2016

A Novel Genetic Algorithm and its Application to Optimize Manufacturing Schedules

Arash Nasrollahishirazi
University of Rhode Island, arashshirazi@uri.edu

A NOVEL GENETIC ALGORITHM AND ITS APPLICATION TO OPTIMIZE MANUFACTURING SCHEDULES

BY

ARASH NASROLLAHISHIRAZI

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHISOLOPHY

IN

INDUSTRIAL AND SYSTEMS ENGINEERING

UNIVERSITY OF RHODE ISLAND

2016

DOCTOR OF PHILOSOPHY DISSERTATION

OF

ARASH NASROLLAHISHIRAZI

APPROVED:

Dissertation Committee:

Major Professor — Manbir S. Sodhi

Gregory B. Jones

Mercedes A. Rivero-Hudec

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2016

ABSTRACT

John Holland and his colleagues at the University of Michigan introduced genetic algorithms (GAs) in 1992. The algorithmic coding of Genetic Algorithms was described by Goldberg. Since of the real world problems studied in operations research and management science are too complex to be solved by using conventional optimization techniques, genetic algorithms have been widely used in their solution. Furthermore, problems with stochastic characteristics are also typical in analysis, design, and operation of modern systems. Stochastic optimization methods are even more complex than deterministic methods. Since the 1960's researches have tried to simulate biological process for solving hard optimization problems including stochastic optimization problems. Evolutionary Algorithms (EAs) have been introduced to imitate natural evolutionary processes of human beings. The Genetic Algorithm is an example of EAs. Simulated Annealing and Genetic Algorithms are two examples of optimization methods applied to solving stochastic optimization problems.

According to Darwin Evolution's theory, a process must exist which determines how traits get passed from one generation to the next. Moreover, there must always be diversity of traits present in the population. The last element of Darwin's Theory is natural selection, which is a way to protect the functional advantages that enables a species have anadvantage in competition with others in nature. John Holland used these ideas from Darwin's Theory when he introduced the Genetic Algorithm. GAs initiate with a set of random solutions for the problem, and this set of solutions is called the population. Each individual (random solution) in a population is called a chromosome and satisfies the constraints for the problem. To facilitate convergence and make the algorithm less sensitive to modeling error, randomness is occasionally used in the search process. This dissertation, which proposes, tests and utilizes a new approach for GAs, will be discussed in four manuscripts:

The objective of **Manuscript I** (*in preparation for submission to the journal of Association for Computation Machinery*) was to modify the conventional concept of Genetic Algorithm base on human cell division mechanisms. Based on an undirected mechanism of evolution and the natural selection processes, genetic algorithms have been applied for solving many complex problems. Generally, GAs work with a pool of candidate solutions (codified as a genome expression) via crossover and mutation mechanisms for generating new solution proposals for the problem. Algorithms differ in the customization of the genome representation of the solution for the problem, and in the fitness function used to evaluate the quality of solutions, based on the problem characteristics. In this paper an extension of the genetic algorithm itself is described. Using correspondents to the Mitosis and Meiosis processes for cell division, a framework for an extended genetic algorithm is developed. Numerical results with benchmark problems show that the solution quality obtained using the proposed algorithm is superior to that achieved by application of the original Genetic Algorithm. However, proposed GA couldn't intelligently control the populations for the rate of Meiosis and Mitosis.

Manuscript II (*in preparation for submission to Journal of Production Research*) presents an intelligent controller to increase the performance of proposed GA and tested on the flow line sequencing problem to check the robustness of algorithm on real manufacturing problem. This part focuses on the general form of the flow line scheduling problem with the objective of minimizing the makespan. Fuzzy Cell Genetic Algorithm (FCGA) algorithm makes use of fuzzy logic to control the cell mechanisms intelligently, applied for flow line sequencing problems. This approach is

intended to improve the performance of genetic algorithm in permutation flow line scheduling problems. A relative evaluation of the FCGA with well-known existing heuristic and metaheuristic methods on recognized benchmarks problems is presented. FCGA is found to be very efficient on examined problems in comparison with other algorithms which solved in permutation schedules for the problems.

In **Manuscript III** (*in preparation for submission to Journal of Operation Research*), the novel algorithm (FCGA) has been tested on Flexible flow line problems, a more complex version of the flow line problem, to minimize the makespan for the process. Real world applications of this problem can commonly be found in printing and electronic circuit board manufacturing industries. A generalized integer programming (IP) model for this problem is proposed. The Fuzzy Cell Genetic Algorithm (FCGA) is proposed to solve the IP model, which has been proven to be NP-hard. Sample problems are generated with known good solutions to evaluate the effectiveness of the FCGA approach. The FCGA matches the performance of the IP model for small sized problem instances and it is proven to be effective for larger problem instances. The results show the robustness of FCGA for flexible flow line problems.

Manuscript IV, *(in preparation for submission to international of Production research)* focuses on different manufacturing problems which are more specifically related to assembly line balancing problem. It is not simple to solve this class of manufacturing problem based on just a single objective; hence a multi-objective genetic algorithm has been designed. For automotive assembly, robots have been used to increase line productivity and efficiency, and improve quality. However, robot failures reduce the throughput rate and product quality. There are several ways of

recovering from line failures. One approach is to establish a manual backup station dedicated to processing those jobs that were incomplete when the line failed, allowing the line to re-start with fresh jobs at each station. Operations performed at this station usually take longer, and are not commensurate in quality with the automated stations. Another approach, developed in this paper, is to design a line with some redundancy: in-line backup stations and a manual recovery station. The backup stations are part of the main line but utilize versatile robots. In this case, a line failure is handled by reconfiguring the backup stations to perform as many make-up operations as possible, but the manual backup station is used for tasks that are not very demanding in complexity or precision. A multi-objective Genetic Algorithm approach for line design and for reallocating tasks when failure occurs is presented. This system is compared with an alternate approach that configures the line with a high level of redundancy and uses a backup station for all recovery. A comparable throughput is achieved with lower levels of redundancy and with fewer jobs sent for manual completion.

ACKNOWLEDGEMENT

My doctorial dissertation would not be possible without the assistance of many people. First of all, I appreciate the support of my major professor, Professor Manbir Sodhi, for giving me a chance to work with him as a graduate student. His guidance, patience and personality thought me how to improve my skills and more importantly, the way of thinking. I am thoroughly grateful to learn many things from him in both academia and personal aspects. I believed that I have spent some of the best years of my life under his supervision and look forward to working with him in future.

I would also like to extend my thanks and appreciation to my committee members, Dr. David G. Taggart, Dr. Gregory .B. Jones, Dr. Mercedes A. Rivero-Hudec, and Dr, Hany Al-Ashwal for their time and scientific inputs.

I also thank all my colleagues and office mates at department of Mechanical, Industrial and Systems Engineering for exchanging ideas and helpful suggestions.

Finally, I need to mention that my family never stopped supporting me in all the ways they could. My Mother is the main reason that keeps me going endlessly. Her encouragement and passion makes me work consistently and efficiently. I am also, indebted to my father, brother, and sister for their endless support.

PREFACE

This dissertation is presented in manuscript format in accordance with University of Rhode Island Graduate School Guidelines. This dissertation is composed of four manuscripts that have been combined to satisfy the requirements of the department of Mechanical, Industrial and Systems Engineering.

MANUSCRIPT I: MODIFIED GENETIC ALGORITHM BASED ON HUMAN CELL MECHANISMS

This manuscript was submitted to the Journal of Association Computing Machinery.

MANUSCRIPT II: SCHEDULING FLOW LINE BY FUZZY CELL GENETIC ALGORITHM

This manuscript is in preparation for submission to Soft Computing Journal.

MANUSCRIPT III: FUZZY CELL GENETIC ALGORITHM APPROACH FOR FLEXIBLE FLOW LINE SEQUENCING MODEL

This manuscript is in preparation for submission to International Journal of Operation Research.

MANUSCRIPT IV: MULTI-OBJECTIVE GENETIC ALGORITHM FOR BACKUP STATION IN FAULT-TOLERANT FLOW LINE DESIGN

This manuscript is in preparation for submission to International Journal of Production Research.

TABLE OF CONTENTS

LIST OF TABLES

MANUSCRIPT I

MANUSCRIPT II

MANUSCRIPT III

MANUSCRIPT IV

LIST OF FIGURES

MANUSCRIPT I

MANUSCRIPT II

MANUSCRIPT III

MANUSCRIPT IV

MANUSCRIPT – I: MODIFIED GENETIC ALGORITHM BASED ON HUMAN CELL MECHANISMS

This manuscript is in preparation for submission to Soft Computing Journal.

Arash Nasrolahi Shirazi[1], Meghan Steinhaus[1], and Manbir Sodhi[1]

[1]Department of Mechanical, Industrial and Systems Engineering, University of Rhode Island, Kingston RI 02881

Corresponding author: Arash Nasrolahi Shirazi

Department of Mechanical,

Industrial &Systems Engineering

University of Rhode Island

Kingston, RI 02881

E-Mail: arashshirazi@my.uri.edu

ABSTRACT

Based on an undirected mechanism of evolution and the natural selection processes, genetic algorithms have been applied for solving many complex problems. In general these algorithms work with a pool of candidate solutions (codified as a genome expression) via crossover and mutation mechanisms for generating new solution proposals for the problem. Algorithms differ in the customization of the genome representation of the solution for the problem, and in the fitness function used to evaluate the quality of solutions, based on the problem characteristics. In this paper an extension of the genetic algorithm itself is described. Using correspondents to the Mitosis and Meiosis processes for cell division, a framework for an extended genetic algorithm is developed. Numerical results with benchmark problems show that the solution quality obtained using the proposed algorithm is superior to that achieved by application of the original Genetic Algorithm.

INTRODUCTION

Since the 1960's researchers have tried to simulate biological processes in order to solve real life problems in operations research and management science. Evolutionary Algorithms (EAs) were introduced in order to imitate natural evolutionary processes of nature. The Genetic Algorithm (GA) is one of example of an EA. John Holland and his colleagues at the University of Michigan introduced GAs in 1975 [1]. Following the introduction of the GA, Goldberg proposed the GA in its current form for solving optimization problems involving all types of functions, including some whose properties were not completely understood [2]. Since then, GAs have been used in a wider array of applications in engineering including design, robotics, telecommunications routing optimization, traffic and shipment routing, computer-aided molecular design, gene expression profiling and many more problems. The knowledge of genetic mechanisms in 1990s was still developing, however, currently, Genetic Algorithms are designed to be working based on simple models of genetic propagation and relying mostly on the mutation and crossover operators.

LITERATURE REVIEW

Since the 1960's researchers have tried to simulate biological processes in order to solve real life problems in operations research and management science. Evolutionary Algorithms (EAs) were introduced in order to imitate natural evolutionary processes in nature. The Genetic Algorithm (GA) is one example of EAs. John Holland and his colleagues at the University of Michigan introduced GAs in 1975 [1]. Following the introduction of the GA, Goldberg proposed the GA in its current form for solving optimization problems involving all types of functions,

including some whose properties were not completely understood [2]. Since then, GAs have been used in a wide array of applications in engineering including aerospace engineering [3], astronomy and astrophysics [4], financial markets [5], geophysics [6], material engineering [7], and the list continues.

In conventional GAs, population represents as a set of non ordered individuals and there is not an intelligent method for recombining the chromosomes for generating new population. So, reproduction with unlimited rules reduces the variation inside the pool of population. Ever since, GAs have been received so much attention to make the structure more robust by intelligently form and control the communication among individuals. This communication can be categorized with two main methods: Spatial segregation introduces the heterogeneity into the pool of population and spatial distance measures the individual distance inside population [8]. The spatial segregation model performs as a group of subpopulations. Individuals in each subpopulation evolve separately. However, by considering certain rules individuals allow to migrate from one population to another in defined iterations. Island model is well-known example of spatial segregation. The spatial distance model restricts the mating process to the individual distance. In this process, only the individuals with the close distance are allowed to reproduce. The neighborhood and the Cellular GAs (cGAs) are two examples in this category.

In Island model GAs, a single population in conventional GAs replaced by multiple subpopulations. Each subpopulation referred as an island which performs a GAs processes and searches in a solution space separately. Subpopulations exchange the number of individuals between each other in a process called migration. In

migration, two parameters such as migration interval and migration size have a significant effect on its performance. Migration interval is defining the period between each migration and migration size is representing the rate of individuals to exchange between islands. Island model have been reported as a superior method to search in the solution space and consequently find result with higher quality in compare with GAs with single population [9]. Researchers have mentioned two main reasons for improving the quality of solutions in models with multiple subpopulations. First, subpopulations maintain independency level for exploring different regions in solution space. Second, using migration operation between subpopulation shares the information at the same time to keep the genetic diversity [10, 11]. Recently, Artyushenko [12] studied the performance of island model to find a global optimum by just operating mutation inside subpopulations.

Cellular GAs (cGAs) represents another structure of GAs which has been characterizing by their spatially decentralized population and different policies for updating the individuals. The population structured in a specific topology and there is a restriction for GAs operations to take a place in a small neighborhood of individuals [13]. It has been reported that there is a correlation between performances of the cGAs and shape of the population [14]. To improve exploitation and exploration processes in cGAs recommended making the model adaptive which dynamically reshapes the population [14].

Cellular GAs has been criticizing for its need to tune the parameters such as population size, mutation rate, and number of crossover points [8]. Terrain-Based Genetic Algorithm (TBGA) is a solution for self-tuning version of the traditional

cGAs. In TBGA model, different combinations of parameter values exist in various physical locations of the population [15].

To improve the performance of island model and cGAs should extensively configure the parameters which causes a significant concern for tuning the parameters to operate optimum [16]. To eliminate the parameter control, Dick [17] proposed spatially-dispersed genetic algorithm (sdGA) to improve searching abilities of the cGAs on some problem domains. The sdGA uses x-y Euclidian space to place the population inside it. Reproduction operation just considers the parents with the visible territory inside the space. The new chromosomes situate in a visibility radius as parent chromosomes.

A Multinational Evolutionary Algorithms (MEA) is other types of Evolutionary Algorithms which uses subpopulations to find more quality solutions in global and local optimums [18]. The motivation behind MEA is to distribute subpopulations in the fitness topology landscape and search in particular part of the search space. At the end of the searching process, MEA returns more than just a single solution. The new concepts such as world, nations, governments and politicians have been introduced during the MEA procedures. The MEAs start with grouping the entire individuals under one nation. Later on, new nations may be generated by using reproduction processes between the individuals in the same nation. The rules for migration the individuals between nations are controlled by fitness-topology function known as *hill-vally*. There is a possibility for nations to merge together if they approach the same optimum point.

In 2007, 2008, and 2009, the Social based Model GA (SBGA) and Human Community Base GA (HCBGA) models proposed by inspiring the real world community and human social interactions [19-21]. The two models follow the same path as island model but each island represents different communities in the world. Each individual has different attributes such as sex, age, and social level in society. The mating process is based on the natural and social selection in human societies. There is more restriction in this model which two individuals in the same family cannot generate new population.

The latest structure of island model could find in Reza et al.[22] research which refereed as Multilevel Cooperate Genetic Algorithm (MLGA). The MLGA consist of subpopulations which each one is divided in many groups. The evolution procedures occur in two levels such as individuals and group level. At the individual level, the procedures of mutation and crossover and selection of chromosomes are applied to the individuals in groups. However, at the group level, colonization operator is happening in which an eliminated group in a systems replaced by offspring of a colonist group. The robustness and effectiveness of MLGA has been shown by experimental results on well-known optimization functions.

GENETIC ALGORITHM

Inspiration

In 1992, John H. Holland provided his inspiration about GAs, and how he used natural selection theory with sexual reproduction in an organism. Natural selection

dictates which individual in the population is dominant to reproduce, and the sexual reproduction creates new offspring by sperm and ova fuse from dominate individuals. Natural selection is the last element of Darwin's Theory, which serves as a way to protect the functional advantages that enable a species to have a privilege over others in nature. Holland pursued the idea of natural selection further, in order to find the genetic reason behind the successiveness of individuals. He found sexual reproduction as the logic behind evolution. Homologous recombination (crossover) and mutation are the main actions in sexual reproduction. These actions serve as the key to improving DNA through evolution. Crossover is a type of genetic recombination in which genetic material is exchanged between two homologous chromosomes. Mutation is the next genetic phenomena in sexual reproduction. This is a permanent change of the sequence of information in genome. There is no guarantee that the DNA of an organism will improve after mutation. All of these biological operations are fundamental procedures in the GA. The biological operations used in the GA have made it simple to implement and outstanding for solving different problems across various fields. However, there are two questions left here. First, is the GA following all the mechanisms in human cells during the natural selection? Second, is it practical and beneficial for the GA to implement more of the main human cells procedures? This paper will answer these two questions in the following sections.

Human cell division

This part will explain the two main processes in human cell division. Cell division consists of two main mechanisms: Mitosis and Meiosis (Fig. 1). During mitosis process, a cell with 2n number of chromosomes starts growing and

chromosomes aligning. Then, chromatids (one of two identical chromosomal strands) move toward the left or right side of the cell without any physical connection. Finally, the content of the cell is divided into two new daughter cells, and the chromosomes in each are replicated to two sister chromatids. The final product of mitosis is two new cells, containing 2n chromosomes. Meiosis consists of two sub-steps in the process namely Meiosis I and Meiosis II. The final product of this process includes four cells where each one containing two distinct chromatids. Crossover happens during Meiosis I. The final step in Meiosis occurs when four cells and each one contains two distinct chromatids. Mitosis and meiosis are two distinct processes in cell division for making sexual and asexual chromosomes.

IMPROVING GAs BY USING SYSTEMATIC BIOLOGICAL PROCESS IN HUMAN CELLS

The main objective of this paper is to develop modified version of GAs that is includes by mitosis and meiosis mechanisms in human cell division.

In conventional GAs, the only actions are mutation and crossover, where these partially follow the meiosis process for sexual chromosomes. However, the genetic algorithm proposed here follows the previously explained human cell mechanisms more closely. Mitosis and meiosis operations are known to be immensely important in cell reproduction. Therefore, these two processes will be used in an algorithmic way. The central hypothesis underlying this paper is that since cell division works based on a defined mechanism including multiple steps, incorporating these steps into the GA

will result in better quality solutions than the classic GA. This could lead to an optimization approach for finding possible solutions to complicated problems in multiple steps. Cell growth, mitosis and meiosis are three steps in the cell cycle that will be covered in this paper.

It is important to note that, not all the mechanisms that nature uses prosperously provide the best solutions for optimization problems. But, the proposed GA has some characteristics which have shown improvements on evolutionary algorithms. Decentralizing is one the characteristic which partitions the population in to several subpopulations [23-24]. Decentralizing methods improved the search in solution space and promote the numerical and runtime of the algorithm [25-29]. Migration of individuals is another specification of the proposed GA which have been proved to enhance the quality of solutions [24],[30].

Conventional GAs

In Genetic Algorithms, a single chromosome represents a solution for problem. These chromosomes can be represented by a string of symbols, numbers or binary bit strings. A fitness function is used for evaluating the quality of each chromosome.

Mechanistically, new solutions or chromosomes (offspring) should iteratively evolve through cell processes in two steps. The first process is Crossover. It creates an offspring by merging the information in two chromosomes (parent chromosomes) from the current population. The second process is mutation, which modifies the offspring; which could result in either a better or a worse solution. The role of randomness in mutation is to find areas of the search space that may contain

unexpectedly near optimal solutions. The possibility of mutations happening is very small. After performing cell processes, the offspring group can make a new population or a new generation. According to the theory of natural selection, the parent chromosome with a lower fitness value can be replaced by the offspring; therefore the size of the new population will remain constant. Michalewicz generalized GAs as consisting of five basic steps [30]:

1. Generating feasible random solutions for the problem.
2. Perform genetic processes (crossover and mutation) for creating new offspring.
3. Evaluate all chromosomes with the fitness function.
4. Generate a new population by selecting the offspring and the old parent chromosomes.
5. Terminate the process if the results are acceptable for the decision maker, otherwise go to step2.

Proposed GA based on human cell

The proposed GA mimics mitosis and meiosis in the human cell. Previously, conventional GAs mimicked the meiosis processes on sexual chromosomes. In the proposed GA, however, the concept of mitosis for the asexual chromosomes will be added to GA. Fig. 2 shows the flowchart for the proposed GA. The proposed GA starts by generating chromosomes for the populations (population1 and population 2). They are shown as N_1 and N_2 with the size of n_1 and n_2. Chromosome generation is a completely random process, and the chromosomes for each population are created separately.

After generating populations, the parameters are set. The definition of parameters such as m_1, m_2, α_1, α_2, β_1, β_2, and the range of them are introduced in Table I. The parameters are important as they mix the chromosomes from population1 and population2 to make sub-population1 and sub-population2.

The two subpopulations are shown as N'_1 and N'_2. The sizes of subpopulations are formulated as follows:

$$n'_1 = \alpha_1 . m_1 . n_1 + \beta_1 . m_1 . n_2 \qquad (1)$$

$$n'_2 = \alpha_2 . m_2 . n_1 + \beta_2 . m_2 . n_2 \qquad (2)$$

In formulations (1) and (2), different number of chromosomes from population 1 and population 2 merged for performing the meiosis and mitosis procedures. The main purpose of the parameters is to define how the algorithm collects asexual chromosomes and sexual chromosomes in to the subpopulation 1 and subpopulation 2 for mitosis and meiosis purposes.

A group of asexual chromosomes in N′1 undergoes mitosis, which consists of duplication and mutation actions. At the same time, the N′2 which consists of sexual chromosomes undergoes homologous recombination or crossover in meiosis phase I. There is a possibility of mutation in meiosis phase II. Normally, there is not a high rate of mutation in the mitosis and the meiosis processes.

In genetics, there are different factors such as “exogenous” (environmental factors) and “endogenous” (errors during DNA replications) which may cause

mutations. Mutations occur randomly. The randomness in mutations can have no effect, change the product of a gene, or prevent the gene from functioning properly. In the proposed algorithm, mutations have a similar concept of randomness but the mutations in mitosis and meiosis are of different forms. The rate of mutations in mitosis is chosen to be higher than meiosis. At the end, after mutation in mitosis and meiosis, the successive chromosomes in mitosis are going to N_1. Also, the chromosomes at the end of meiosis are going to N_2 based on elite characteristics of the chromosomes. Both mitosis and meiosis execute at the same time in parallel.

RESULTS AND ANALYSIS

The existing GA and the proposed GA were tested on the eight nonlinear benchmark functions [31]. The benchmark functions are consisting unimodal and multimodal optimization problems. The Rosenbrock, Sphere, and Exponential functions are categorizing as unimodal landscape functions and the remaining are the multimodal functions. Please note that benchmark functions are minimization problems. Each algorithm was used to solve each benchmark functions of 100 times to eliminate random discrepancy.

Before comparing the performance of GA and proposed GA on all benchmark functions, the quality of solutions and the completion time for two algorithms is evaluated on the Rosenbrock function. The global optimum in Rosenbrock lies inside a long, narrow, parabolically-shaped flat valley. It is trivial for an algorithm to find

the valley. However, finding the global optimum within the valley is extremely difficult.

The length of chromosome is represented by D. In the first part, D is equal to 20. To compare the two algorithms, the total population sizes for each are set equal to each other. In other words, if the population size in the GA is 100, this is the total number of chromosomes (50 in population1 and 50 in population2) in the proposed GA. Here, the probability of mutation (p_m) and probability of crossover (p_c) have the same concept as Goldenberg probabilities which are explained in his book [2]. The fixed values for crossover and mutation probabilities have been chosen according to the best performance of GA in a previous study [33]. The new concept of the probability of mutation in mitosis is shown with p'_m. It follows the same rules as p_m and p_c.

The existing GA and the proposed GA are tested in a comparative setting with 3 different population settings. Each case has been replicated 100 times. Neither the existing GA nor the proposed GA is able to find the optimum point across all 100 replications (Table III). The comparison in Table III provides the final solutions for the Rosenbrock function found by both algorithms. The superior algorithm is the one with the closer results to the optimum point.

In all of the tested cases, the proposed GA finds better results compared to the existing GA. Both algorithms were run with parallelized code.

In second part of the results, the performance of GA and Proposed GA evaluate on other unimodal and multimodal benchmark functions which shown in

Table II. In this section, D is equal to 5, 20, and 30. The population size is fixed in all cases. Each dimension has been replicated 100 times. All the test results in terms of the mean final best and its standard deviation are presented in Table IV. The winner algorithm is the one with the closer results to the optimum point which highlighted in boldface.

The Sphere function is not a complicated landscape for proposed GA but for classic GA it is difficult to get close to the optimum solution. However, GA and proposed GA provides near optimal solutions for each problem size which is tested with the Exponential and Griewank functions. The Rastrigin function categorized as a complex problem because it is highly multimodal. In regards to the Rastrigin function, proposed GA performs much better that GA on dimensions of 5, 20, and 30. The 2n minima function has local optimum points and global optimum which located on it flat button. GA and proposed GA present the solution close to the global optimum with the chromosome with size 5. However, proposed GA shows much better results than GA for bigger chromosomes sizes (20 and 30). GA traps in local minimum for the Ackely function but proposed GA provides near optimal solution for the problem in all dimensions.

Fig. 3 shows the performance of GA and proposed GA after 100 iterations for the chromosome with size 5. The Schwefel function is the last benchmark function. It has a second best minimum solution far from the global optimum, which makes the function difficult for algorithms to find the global minimum too. Proposed GA shows superior results in comparison with GA for the Schwefel function with the dimension sizes 5, 20, and 30.

By considering the combination of benchmark functions and the size of the dimensions, we have 24 test cases. In all cases proposed GA yields the best solutions and it shows the human cell processes enhances the exploration of the search space effectively.

DISCUSSION

The relative simplicity of how the GA is inspired by the natural processes of genetics has resulted in it becoming a popular approach to find the globally optimal solution (best solution) among many locally optimal solutions. By delving deeper into genetics, however, it is evident that there are more sophisticated processes involved in human cell divisions than those are being used in the original GA. Including more of the genetic processes into the proposed algorithm could result in superior performance on the benchmark functions. According to the results in Table III, the proposed GA has an improved running time in all three cases by approximately 42-45% over the original GA. Although this reduction in run time is trivial for simple problems, the mean solution values in all cases show that the proposed GA works superior to the original GA in solution quality.

The performance of both algorithms on Rosenbrock function, without changing the population size, the original GA was only able to improve the mean solution value by 1.8%; whereas the proposed GA was able to improve the mean value by nearly 39%.

The reason behind this progress is that proposed GA is less likely gets stuck in a local optimum. The proposed GA conducts a more robust search of the solution space.

Increasing the population size from 100 (Case I and Case II) to 500 (Case III) has a significant effect on the best solution found by both the original GA and the proposed GA. This fact is not, however, a significant indicator of the performance of either algorithm. Increasing the population size of any GA should improve the algorithm's performance, since a larger pool of random solutions will certainly provide a more diverse search of the solution space.

Although it is tempting to increase the population size to ensure a better quality solution, it is important to note that an increase in the population size will also increase the algorithm's running time, as seen in the results of Table III.

However, using only one benchmark problem is not sufficient to draw any firm conclusion when comparing evolutionary algorithms. Therefore, simple GA and Proposed GA have been tested on additional benchmark problems with different sizes to provide strong conclusion. As it is illustrated in Table IV, the proposed GA has significantly better results than GA on Sphere and Schwefel functions with the chromosomes size 5. However, with the same size, both algorithms show approximately the same results on other benchmark functions. For the larger chromosome sizes (20, 30), proposed GA provide superior solutions for the all test functions in compare with simple GA. The GA stagnation problem can be clearly observed for the Ackley function in Table IV. The conventional GAs are trapped in

local optima for all dimension sizes, while the proposed GA searches continuously to get closer to the global optimum.

As it is shown in Table III and Table IV, there are two major advantages associated with the modified version of GA over the classical one, including the higher quality and shorter computational time in all three cases. It should be mentioned that regardless of the parameter settings, the proposed algorithm has superior performance.

From the time that Holland first proposed the original GA to now, there have been numerous modifications proposed to the GA in order to improve the computational time and solution quality. The purpose of this paper is to perform a comparative investigation between the original GA developed by Holland and the proposed GA which incorporates the real cell division processes in human genetics, therefore, only the original GA is considered. Since the proposed GA has the same, basic structure of the original GA, many of the improvements and modifications that have been proposed to Holland's GA can also be applied to the proposed GA.

CONCLUSION

Researchers working on a broad range of scientific fields have been using the genetic algorithm to solve complex problems for many years. Many of these researchers found GA as a successful tool when other methods failed. Because of the nature of GA, it is extremely easy to implement.

This could be a major reason that makes the GA as a flexible tool to be applied to many problems. The proposed GA does not change the nature of GA, nor does it make the GA more complex to implement.

The proposed algorithm implements an additional process from genetics within the GA, and is able to improve both solution quality and computational time. This research illustrates that there is still a significant potential for employing emerging knowledge about genetic processes to enhance computer programs in order to solve complex problems.

It should be mention that this research does not indent to introduce a new evolutionary algorithm which can compete with recent methods. It is intent to demonstrate genetic propagation to prove that how the performance of GA will change by using human cell mechanisms. However, proposed GA has a great capability of hybridization which can compete with the most recent metaheuristic methods for combinatorial optimization problems.

ACKNOWLEDGMENT

This work was partially supported by University Of Rhode Island.

REFERENCES

[1] H. John “Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial inteliigence” Oxford, England: U Michigan Press. vol. viii, pp. 183, 1975. (references)

[2] D. E. Goldberg “Genetic algorithms in search, optimization, and machine learning”.Reading, MA: Addison-Wesley, 1989.

[3] Obayashi, S., Sasaki, D., Takeguchi, Y. and Hirose, N. Multiobjective evolutionary computation for supersonic wing-shape optimization. IEEE transactions on evolutionary computation, 4(2), pp.182-187,2000

[4] Charbonneau, P. Genetic algorithms in astronomy and astrophysics. The Astrophysical Journal Supplement Series, 101, p.309,1995.

[5] Mahfoud, S. and Mani, G. Financial forecasting using genetic algorithms. Applied Artificial Intelligence, 10(6), pp.543-566, 1996.

[6] Sambridge, M. and Gallagher, K. Earthquake hypocenter location using genetic algorithms. Bulletin of the Seismological Society of America, 83(5), pp.1467-1491, 1993.

[7] Giro, R., Cyrillo, M. and Galvao, D.S. Designing conducting polymers using genetic algorithms. Chemical Physics Letters, 366(1), pp.170-175, 2002.

[8] Lim, T.Y. Structured population genetic algorithms: a literature survey. Artificial Intelligence Review, 41(3), pp.385-399, 2014.

[9] Muhlenbein, H. Evolution in time and space-the parallel genetic algorithm. In Foundations of genetic algorithms, 1991.

[10] Starkweather, T., Whitley, D. and Mathias, K. Optimization using distributed genetic algorithms. In International Conference on Parallel Problem Solving from Nature (pp. 176-185). Springer Berlin Heidelberg, 1990.

[11] Whitley, D., Rana, S., & Heckendorn, R. B. The island model genetic algorithm: On separability, population size and convergence. Journal of Computing and Information Technology, 7, 33-48, 1999.

[12] Artyushkenko B. Analysis of global exploration of island model genetic algorithm. CAD systems in microelectronics, 2009. CADSM 2009. 10th international conference, pp 280–281, 2009.

[13] Alba, E., Alfonso, H., & Dorronsoro, B. Advanced models of cellular genetic algorithms evaluated on SAT. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation* (pp. 1123-1130). ACM, 2005.

[14] Alba, E., & Dorronsoro, B. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation*, *9*(2), 126-142,2005.

[15] Gordon, V. S., Pirie, R., Wachter, A., & Sharp, S. Terrain-based genetic algorithm (TBGA): Modeling parameter space as terrain. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1* (pp. 229-235). Morgan Kaufmann Publishers Inc, 1999.

[16] Cantú-Paz, E. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, *10*(2), 141-171, 1998.

[17] Dick G. The spatially-dispersed genetic algorithms. Lecture notes in computer science, 2003, vol 2724, genetic and evolutionary computation—GECCO 2003, p 210,2003.

[18] Ursem RK. Multinational evolutionary algorithms. In: Proceedings of the congress of evolutionary computation , vol 3. IEEE Press, pp 1633–1640, 1999.

[19] Al-Madi NA, Khader AT (2007) A social based model for genetic algorithms. In: Proceedings of the third international conference on information technology (ICIT), 9–11 May 2007, Zaytoonah University, Amman.

[20] Al-Madi NA, Khader AT. De Jong's sphere model test for a social-based genetic algorithm (SBGA). IJCSNS Int J Comput Sci Netw Secur 8(3):179–187, 2008.

[21] Al-MadiNAQ. Ahuman community-based genetic algorithm model (HCBGA).Universiti SainsMalaysia (USM), Ph.D. Dissertation, 2009.

[22] Akbari, R., Zeighami, V., & Ziarati, K. Mlga: A multilevel cooperative genetic algorithm. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on* (pp. 271-277). IEEE, 2010.

[23] E Alba and M Tomassini "Parallelism and Evolutionary Algorithms", IEEE Transations on Evolutionary Computation, vol. 6, no. 5, 2002.

[24] T. C. Belding, "The distributed genetic algorithm revisited," in Proc. 6th Int. Conf. Genetic Algorithms, L. J. Eshelman, Ed., 1995, pp. 114–121.

[25] ___,"Distributed genetic algorithms," in ICGA-3, J. D. Schaffer, Ed., 1989, pp. 434–439.

[26] S. Baluja, “Structure and performance of fine-grain parallelism in genetic search,” in Proc. 5th Int. Conf. Genetic Algorithms, S. Forrest, Ed.,1993, pp. 155–162.

[27] ___,“Improving flexibility and efficiency by adding parallelism to genetic algorithms,” Statist. Comput., vol. 12, no. 2, pp. 91–114, 2002.

[28] V. S. Gordon and D. Whitley, “Serial and parallel genetic algorithms as function optimizers,” in Proc. 5th Int. Conf. Genetic Algorithms, S. Forrest, Ed., 1993, pp. 177–183.

[29] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer Science & Business Media, 1996.

[30] Pornsing, Choosak, Manbir S. Sodhi, and Bernard F. Lamond. "Novel self-adaptive particle swarm optimization methods." Soft Computing (2015): 1-15.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE Trans. Evol. Comput., vol. 6, pp.182 -197, 2002 .

[32] Srinivas, M., and Lalit M. Patnaik. "Adaptive probabilities of crossover and mutation in genetic algorithms." Systems, Man and Cybernetics, IEEE Transactions on 24, vol. 4, pp. 656-667, 1994.

TABELS

TABLE 1. PARAMETERS FOR PROPOSED GA

Parameter	Definition	Number
m_1	Mitosis rate	[0 , 1]
m_2	Meiosis rate	[0 , 1]
$\alpha_1 . m_1$	ratio of chromosomes in N_1 that should place in N'_1	[0 , 1]
$\alpha_2 . m_2$	ratio of chromosomes in N_1 that should place in N'_2	[0 , 1]
$\beta_1 . m_1$	ratio of chromosomes in N_2 that should place in N'_1	[0 , 1]
$\beta_2 . m_2$	ratio of chromosomes in N_2 that should place in N'_2	[0 , 1]

TABLE 2. RESULTS FOR GA AND PROPOSED GA ON ROSENBROCK FUNCTION

Case	Algorithm	Iteration	N	N'	p_c	p_m	p'_m	Mean	Best Solution	Average time(Sec)
I	GA	***100***	***100***	***0***	***0.5***	***0.001***	***0***	***296.931***	***131.690***	***0.218***
	Proposed GA	***100***	***50***	***50***	***0.5***	***0.001***	***0.16***	***47.539***	***0.810***	***0.126***
II	GA	***500***	***100***	***0***	***0.5***	***0.001***	***0***	***291.531***	***122.993***	***0.979***
	Proposed GA	***500***	***50***	***50***	***0.5***	***0.001***	***0.16***	***29.010***	***0.011***	***0.531***
I	GA	***500***	***500***	***0***	***0.5***	***0.001***	***0***	***77.595***	***39.286***	***4.540***
	Proposed GA	***500***	***250***	***250***	***0.5***	***0.001***	***0.16***	***18.075***	***0.006***	***2.491***

TABLE 3. BENCHMARK FUNCTIONS

Name	*Function*	*Range*	x^*	$f(x^*)$
Rosenbrock	$f(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	[-2.048, 2.048] ^{D}	[1,..,1] ^{D}	0
Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$	[-100, 100] ^{D}	[0,..,0] ^{D}	0
Exponential	$f(x) = -exp\ (-0.5 \sum_{i=1}^{D} x_i^2)$	[-1, 1] ^{D}	[0,..,0] ^{D}	-1
Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos\ (\frac{x_i}{\sqrt{i}}) + 1$	[-600, 600] ^{D}	[0,..,0] ^{D}	0
Rastrigin	$f(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10$	[-5.12, 5.12] ^{D}	[0,..,0] ^{D}	0
2^n minima	$f(x) = \sum_{i=1}^{D} (x_i^4 - 16x_i^2 + 5x_i)$	[-5, 5] ^{D}	[-2.90,..,-2.90] ^{D}	-78.33 ^{D}
Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	[-32.768, 32.768] ^{D}	[0,..,0] ^{D}	0
Schwefel	$f(x) = 418.9829\ D - \sum_{i=1}^{D} (x_i\ sin\sqrt{\lvert x_i \rvert})$	[-500, 500] ^{D}	[420.9687,..,420.9687] ^{D}	0

TABLE4 . MEAN FITNESS AND ITS STANDARD DEVIATION OF GA AND PROPOSED GA ON BENCHMARK FUNCTION

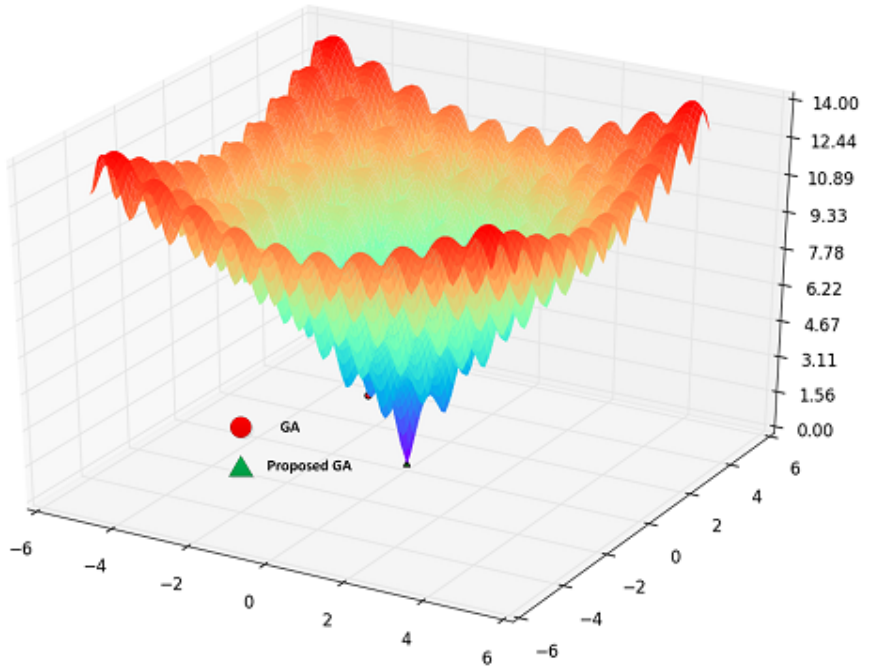
Function	***Dimension***	***GA***	***Proposed GA***
Sphere	5	158.2214 (107.0153)	**0.0048** (0.0074)
	20	5033.8991 (1618.8810)	**0.0238** (0.0293)
	30	11229.5322 (2965.0450)	**0.0641** (0.1279)
Exponential	5	- 0.9895 (0.0085)	**-1** (0)
	20	- 0.7331 (0.0723)	**-1** (0)
	30	- 0.5076 (0.0860)	**-1** (0)
Griewank	5	0.0731 (0.0562)	**0.0468** (0.0306)
	20	0.7635 (0.1060)	**0.1163** (0.1009)
	30	0.9402 (0.0507)	**0.1571** (0.1444)
Rastrigin	5	8.1607 (3.1907)	**0.0030** (0.0041)
	20	90.1284 (17.1666)	**0.0120** (0.0159)
	30	179.4350 (26.4691)	**0.0262** (0.0379)
2^n minima	5	- 380.1760 (10.5791)	**- 391.6613** (0.0005)
	20	- 1306.7493 (61.2300)	**- 1566.6445** (0.0050)
	30	- 1824.0181 (94.1857)	**- 2349.9656** (0.0059)
Ackley	5	1.7183 (0)	**0.0471** (0.0362)
	20	1.7183 (0)	**0.0435** (0.0343)

TABLE4 . MEAN FITNESS AND ITS STANDARD DEVIATION OF GA AND PROPOSED GA ON BENCHMARK FUNCTION

Function	***Dimension***	***GA***	***Proposed GA***
	30	1.7183 (0)	**0.0623** (0.0648)
Schwefel	5	63.6211 (47.6583)	**0.0155** (0.0212)
	20	1203.3818 (340.5058)	**0.0744** (0.1003)
	30	2681.7815 (484.3382)	**0.2057** (0.3003)

FIGURES

FIGURE 1. Comparison of GA and proposed GA in a search space on Ackley's benchmark function in iteration 100 with the problem size 5.



MANUSCRIPT – II: SCHEDULING FLOW LINE BY FUZZY CELL GENETIC ALGORITHM

In preparation for submission to International Journal of Production Research

Arash Nasrolahi Shirazi[1], Meghan Steinhaus[1], and Manbir Sodhi[1]

[1]Department of Mechanical, Industrial and Systems Engineering, University of Rhode Island, Kingston RI 02881

Corresponding author: Arash Nasrolahi Shirazi

Department of Mechanical,

Industrial &Systems Engineering

University of Rhode Island

Kingston, RI 02881

E-Mail: arashshirazi@uri.edu

ABSTRACT

This paper focuses on the flow line scheduling problem with the objective of minimizing the makespan. An integer program that incorporates this aspect of the problem is formulated and discussed. Because of the difficulty of solving complex problems using the IP directly, a fuzzy control Genetic Algorithm (FCGA) used in conjunction with a genetic algorithm mimics the Mitosis and Meiosis mimics of in human cell reproduction. This approach is used to solve permutation flow line scheduling problems. A relative evaluation of the FCGA with well-known existing heuristic and metaheuristic methods on recognized benchmarks problems in this domain is presented. FCGA found to be very efficient on due examined problems in comparison with other algorithms used to solve these problems.

Keywords: permutation flow line, makespan, fuzzy cell genetic algorithm, mitosis, meiosis, fuzzy logic, metaheuristic, non-permutation.

INTRODUCTION

Manufacturing systems are configured in many ways. One of the most famous instances is flow line system which all jobs must be processed on all machines with the same sequence. Transfer lines, assembly line, chemical plants, and logistics are some of the example of scheduling flow lines problems [1]. Nawaz [2] described the flow line sequencing problems as that of processing a set of m jobs with the same specific order performed from one machine to another. Therefore, each job visits all n machines with the same order. The main objective is to minimize the processing time of the job with the highest order at the n^{th} machine in the system. The flow line sequencing problem can be categorized in to permutation and non-permutation flow line cases (Fig.1). The sequence of each job in permutation flow line remains unchanged, however in non-permutation flow line the sequence of jobs can be changed on each machine. It has been long time that the researchers have focused on permutation schedules due to relative combinatorial simplicity of schedules that can be specified by giving a permutation of the jobs [3]. This paper focuses on permutation model with no buffer or no-wait conditions. Most flow line scheduling models have been known as NP-hard problems except the case where numbers of jobs or machines are limited to 3 or less [4, 5]. It should be noted that the complexity of permutation problems is $n!$ with different schedules for job orders on machines, but non-permutation problems are much more complex, of the order of $n!^m$ (for n number of jobs and m machines)[6]. It has been shown that flow line problems with high instances are not just difficult to solve as integer program but also as heuristics difficult to solve [7].

LITERATURE

There are different approaches for solving flow line sequencing problems (FLP). Branch-and-bound is common technique to determine the optimal makespan solution for permutation schedule problems. The branch-and-bound approach applied for even more complex problem such as flexible flow line model with number of stages and medium buffers in the case of creating non-permutation schedule model or schedules with ideal time in the system [8]. Since FLP problems are considered to be NP-Hard, it is very important to look for heuristic methods that produce an acceptable schedule in practical time. In 1954, Johnson proposed heuristic method to determine optimal solution for special case of two and three machines [9]. Since then, there have been numerous reports by researchers that present different heuristic and metaheuristic approaches over the time. Nawazi et al. proposed NEH heuristic for solving flow line scheduling problems with minimum makespan [2], and this is one of the best known heuristics for the FPS.

The main idea behind NEH is to put the jobs with the larger total processing time in all stages at the first order in the sequencing of the schedule. Taillard [10], Ruiz et.al. [11], and Quan-ke et.al. [12] considered NEH as a best heuristic with makespan criteria. Recently, Singhal et.al. [13] proposed an Improved Heuristic by modifying the NEH algorithm. They stated that their algorithm is faster than NEH to and finds a comparable or better solution with the same complexity as NEH.

The FPS has been solved using not just heuristic but also metaheuristic class of computation techniques such as Tabu search, Simulated annealing, Immune algorithm,

and Genetic algorithms (GAs) [14-17]. Gen (1994) proposed GAs as an efficient technique for solving large scale flow line scheduling problem in comparison with Branch-and-Bound algorithm [18]. In 1995, Colin et. al. studied the performance of GAs on the n-jobs, m-machines permutation flow line sequencing problems to judge against naive Neighbourhood search techniques and a modified simulated annealing algorithm. The study showed that for the small size problems using simple algorithms like naive Neighbourhood search techniques is sufficient but for the large size problems GAs performs relatively better than simulated anneal and naive neighbouring methods [19]. Ruiz et. al. [20] proposed two robust genetic algorithms approaches which they have used novel genetic procedures in operators, hybridization with local search and generating initial population. The two modified GAs approaches were found to be most efficient against 11 other methods including simple GAs, tabu search, simulated annealing, and other advanced techniques on Taillard's well known standard benchmark problems for the FSP [21].

The other class of flow line scheduling which focused on non-permutation flow line scheduling (NPFS) has attracted some attention is to verify the performance in compare with PFS on benchmark problems. Tandon et. al. [22] used the makespan as an objective to compare the performance of NPFS and PFS. They applied enumerational search techniques for small problems and simulated annealing for large problems. In their computational experiment with the benchmark problems, non-permutation schedules here, not surprisingly found to be more effective than the permutation schedules. Rossi et. al [6] explored the effectiveness of ant colony optimization for a flow line with buffers as non-permutation flow line and compare it

with other non-native and native approaches [15],[24-27], metaheuristic algorithms which were initiated by feasible solutions attained by other heuristic or metaheuristic, on Taillard [21] and Demirkol [23] benchmarks. They found the proposed methods had the best performance specifically for the large size problems.

PERMUTATION FLOW LINE SCHEDULING PROBLEM

Johnson [9] proposed the method which could find the optimal solution for the permutation flow shop in a case of two-machine in polynomial time and three-machine in non-polynomial time [29-31]. However, the first mathematical formulation in the field of scheduling was presented by Bellman [28]. Later on, Ignall and Schrage [32] proposed a Branch and Bound approach to find the optimal solution for the small size flow shop problems. Momnicki [33] and Mamahon and Burton [34] applied branch and bound method to find the exact solution of the three-machine problem.

Seda [35] derived the integer programming (IP) model for the general form of flow shop problem with n-jobs and m-machines by reducing waiting time (idle time) for each job at the end of the process on each machine.

In this paper, an IP model with no idle time for n-jobs and m-machines is formulated which can easily be set up for solution using the optimization package GAMS (General Algebraic Modeling System) [36].

The problem of using single flow machine for each station can be expressed as an integer programming model. Let *j* be the number of jobs to be schedule at station s. The definitions for this problem presented as:

Indices

j	number of jobs to be schedule
p	position of job in the sequence
s	number of station
mn	last station
nj	job at the last station

Parameters

$Pt_{s,j}$	processing time of job j in station s
$Ct_{s,p}$	completion time of $j$obs in station p and machine s

Decision variable

$x_{i,j}$	1 if job i is performed in position j and 0 otherwise

$$\text{Objective minimize z} \tag{1}$$

s.t.

$$\sum_{p\in P} x_{j,\mathrm{p}} = 1, \qquad \forall\, j \in J \tag{2}$$

$$\sum_{j\in J} x_{j,\mathrm{p}} = 1, \qquad \forall\, p \in P \tag{3}$$

$$Ct_{1,1} - \sum_{j\in J} Pt_{1,j}\, x_{j,1} \geq 0\,, \tag{4}$$

$$Ct_{1,p} - Ct_{1,p-1} - \sum_{j\in J} Pt_{1,j}\, x_{j,p} \geq 0\,, \qquad \forall p \in P \mid p > 1 \tag{5}$$

$$Ct_{s,1} - Ct_{s-1,1} - \sum_{j\in J} Pt_{s,j}\, x_{j,p} \geq 0\,, \qquad \forall s \in S \mid s > 1 \tag{6}$$

$$Ct_{s,p} - Ct_{s,p-1} - \sum_{j\in J} Pt_{s,j}\, x_{j,p} \geq 0\,, \qquad \forall p \in P \mid p > 1 \tag{7}$$

$$Ct_{s,p} - Ct_{s-1,p} - \sum_{j\in J} Pt_{s,j}\, x_{j,p} \geq 0\,, \qquad \forall p \in P \mid p > 1 \tag{8}$$

$$z \geq Ct_{mn,nj}\,, \qquad \forall\, l \in L \tag{9}$$

$$x_{j,p} \in \{0,1\}, \qquad p, j \in \mathrm{J} \tag{10}$$

$$x_{L\,l,t} \in \{0,1\}, \qquad \{\forall l \in L\}, \{\forall\, \mathrm{i,j} \in \mathrm{T} \mid \mathrm{i} \neq \mathrm{j}\} \tag{11}$$

$$x_{l,i,j} = 0, \qquad \{\forall l \in L\}, \{\forall\, \mathrm{i,j} \in \mathrm{T} \mid \mathrm{i} = \mathrm{j}\} \tag{12}$$

We assume each task in each station can be performed at the same processing rate on each machine. Also, each task in the current station must be performed before moving to the next station. The sequences of the jobs are the same from the first station to the last station.

Eq.(1) represents the objective function which is to minimize the makespan, z. Eq.(2) assigns at least one position to each job. Constraint (3) ensures that job j is assigned to one position. Eq.(4) defines the completion time of job j at the first position is greater than or equal to processing time of the first job at the first station. Constraint set (5) makes sure that the completion time of all jobs at the first station should be greater than or equal to the finishing time of each predecessor job plus processing time of successor job. In Eq.(6), finishing time of the first job which assigned to the machine l at the first station should be greater than or equal to the processing time of the first job at the same machine. Constraint set (7) enforce sequencing job j in each station s greater than 1 by putting the finishing time of each job in station s greater than the finishing time of the same task at the station before. Eq.(8) represents the completion time of job in position p is greater than or equal to completion time of the same job in the previous station. Eq.(9) is going to calculate the accumulation the completion time of the last job in final station is greater than or equal to the objective z. Constraint set (10) define $x_{i,j}$ as a binary variable.

Proposed Genetic Algorithm

The proposed GA uses the mitosis and meiosis mechanisms of the human cell reproduction, which has been introduced in previous research [51]. Previously,

conventional GAs mimicked the meiosis processes on sexual chromosomes. In the proposed GA, however, the concept of mitosis for the asexual chromosomes has been be added to GA. Figure 2 shows the flowchart for the proposed GA. The proposed GA starts by generating chromosomes for the populations (population 1 and population 2). They are shown as N_1 and N_2 with the size of n_1 and n_2. Chromosome generation is a completely random process, and the chromosomes for each population are created separately.

After generating populations, the parameters are set. The definition of parameters such as m_1, m_2, α_1, α_2, β_1, β_2, and the range of them are introduced in table 1. The parameters are important as they mix the chromosomes from population 1 and population 2 to make sub-population 1 and sub-population 2.

The two subpopulations are shown as N_1' and N_2'. The sizes of subpopulations are formulated as follows:

$$n_1' = \alpha_1.m_1.n_1 + \beta_1.m_1.n_2 \quad (9)$$

$$n_2' = \alpha_2.m_2.n_1 + \beta_2.m_2.n_2 \quad (10)$$

The main purpose of the parameters is to define how the algorithm collects asexual chromosomes and sexual chromosomes in to the subpopulation 1 and subpopulation 2 for mitosis and meiosis purposes.

A group of asexual chromosomes in N_1' undergoes mitosis, which consists of duplication and mutation actions. At the same time, the N_2' which consists of sexual chromosomes undergoes homologous recombination or crossover in meiosis phase I.

There is a possibility of mutation in meiosis phase II. Normally, there is not a high rate of mutation in the mitosis and the meiosis processes.

In genetics, there are different factors such as “exogenous” (environmental factors) and “endogenous” (errors during DNA replications) which may cause mutations. Mutations occur randomly. The randomness in mutations can have no effect, change the product of a gene, or prevent the gene from functioning properly. In the proposed algorithm, mutations have a similar concept of randomness but the mutations in mitosis and meiosis are of different forms. The rate of mutations in mitosis is chosen to be higher than meiosis. At the end, after mutation in mitosis and meiosis, the successive chromosomes in mitosis are going to N_1'. Also, the chromosomes at the end of meiosis are going to N_2' based on elite characteristics of the chromosomes. Both mitosis and meiosis execute at the same time in parallel.

Use fuzzy controller for FCGA approach

The solution for developing the proposed GAs is to formulate the biological processes in an adaptive way. One of the most effective studies for adaptive approaches is the adaptive parameter setting techniques (Figure 3) [37]. Two major advantages of making GAs adaptive are avoiding the premature convergence and improving the final results [38–40]. Several methods have been used to track different processes in GAs to prevent premature convergence drawback and improve GA performance [41, 42].

Two ways to monitor GAs processes are methods include updated selection and/or crossover operators and optimizing GA parameter settings [43]. Fuzzy logic

controllers have been successfully used for controlling parameters [44,45]. Fuzzy sets are fundamental building blocks of Fuzzy logic. Each set has a continuous membership function as opposed to a binary membership function. In fuzzy logic, the whole interval of real numbers between zero (False) and one (True) are considered to expand logic as a basis for rules of interface. Crisp sets are the usual sets that have been used in daily problems. A fuzzy logic system (FLS) has four main parts:

- Fuzzifier: Collecting input data as a crisp set and translate them in to a fuzzy set using fuzzy linguistic variables, fuzzy linguistic terms and membership functions. Fuzzy linguistic variables are input and output variables of the system. They are either words or sentence but not numerical values. Linguistic variables break down in to linguistic terms. Membership functions are used in fuzzification and defuzzification steps, and they map the crisp values (non-fuzzy values) in to fuzzy linguistic terms and vice versa.
- Fuzzy rules: A fuzzy rule is a simple conditional (IF-THEN) rule to control the output variable.
- Inference engine: Maps a fuzzy input to a fuzzy output by evaluating fuzzy rules, fuzzy logic operators and membership functions.
- Defuzzifier: Converts the final fuzzy result into the crisp value.

Figure 3 demonstrates a general form of a fuzzy logic system and the components in this system. The use of fuzzy logic system for controlling GAs was the solution to very slow speeds and premature convergence of GAs. Xu et al. reported three reasons for there problems:

- Unfit initiation can control parameters for a given problem.
- GA parameters should be fixed although GAs operations are dynamic.
- Complications based on selecting other parameters such as population size and in understanding their influences [46,47].

The fuzzy controller is used right after the chromosomes have been generated, and if selects the sub-populations for meiosis and mitosis adaptively. The controller that is going to presented here has been successfully tested on a Modified self organizing neural network algorithm [51].

Michael (1993) used a dynamic controller with fuzzy logic techniques in order to define the parameters of the GA. He used his algorithm for the inverted pendulum control experimental task [48]. His results were superior when compared to the conventional GAs. For our proposed algorithm, we are going to use Michael's (1993) method to control meiosis and mitosis rates (m_1, m_2).

As it shown in Table 2, the fuzzy part of algorithm starts with three inputs, six parameters for fuzzifier represented by $x_{i,j}$ (i defines as number of inputs and j represents the position of number on x-axis base on increasing order) and four parameters for defuzzifier with notation of $y_{k,j}$ (k shows the number of outputs).

Inputs are metrics for measuring the proposed GAs performance and prevent it from premature convergence. For the input formulations, F_{ave} is defined as the mean of the fitness distribution in the population and F_{best} (best fitness) is the fitness measure of the best individual (lowest fitness value for minimization problems). The worst fitness (F_{worst}) is defined as the worst individual fitness in the population. The

scale for measuring F_{change} (change in a range of fitness) is to calculate $F_{worst} - F_{best}$ in a predefined number of iterations.

The types of inputs are crisp values. It is necessary to convert the crisp values in to fuzzy sets by using membership functions. Triangular membership function is the type of membership function that is using in this study (Figure 4). The number of parameters needed to convert crisp values to fuzzy and vice versa is equal to:

$$(m + n) \times two\ membership\ function\ center\ parameters \quad (11)$$

In equation 13, m is the number of inputs and the total number of outputs is equal to n. For example, for three inputs and two outputs, one should consider 10 parameters for the fuzzifier and defuzzifier sections.

The inputs measure how efficient the proposed GA is converging toward optimal solution. But as long as the program has not start the genetic processes, initiation condition for the fuzzy controller should be provided to return meiosis and mitosis rates for the rest of procedure. Initially, population 1 is evaluated to find two measurements such as F_{ave} , F_{worst}. The last element (F_{change}) starts with guessing number in the range of [0.0, 1.0]. After calculating all inputs, each input (crisp value) should convert to fuzzy values by using the membership function which specified for that input (Figure 5).

Next step is to turn the fuzzy inputs into fuzzy outputs by using fuzzy rules. The rules are IF-THEN methods. The maximum possible number of rules is based on

the number of inputs combinations, and these rules determine the outputs. For example, if there are three inputs, where each has three elements, the number of combinations is equal to 3^3. Therefore, the greatest possible rule for two outputs is 2×3^3 [48]. In this system, both IF and THEN parts of rules are represented as a fuzzy set. IF part consist of fuzzy inputs that have been taken by membership functions. The output of the system is going to define THEN section by reasoning method. The operators such as "AND" and "OR" are assisting to find the output fuzzy. Table 3 shows the applied rules for meiosis part.

For example, if input 1 has just one output (small) with the value of 0.3, input 2 has two input fuzzy (medium and high) with 0.2 and 0.4 values, and input 3 has just one (high) which equals to 0.5 then rules number 4 and 7, should be used which related to outputs fuzzy medium and high respectively. Also, final values for the medium output is *min (0.3, 0.2, 0.5) = 0.2* and high value calculates as *min (0.3, 0.4, 0.5) = 0.3*.

Defuzzifier is the last step for converting output fuzzy into an output crisp values which can be meiosis or mitosis rates for the next procedure. In this part the weighted average strategy should be use for defuzzifier. The first two parameters of $y_{1,1}$, $y_{1,2}$, $y_{2,1}$ and $y_{2,2}$ are related for meiosis and the others are related to mitosis defuzzifier.

During the processing of the code, the metrics are monitored to assess whether the results if the code is going to lead the premature convergence in that case the fuzzy

controller increases the mitosis rate. However, for improving the quality of solution the controller will increase the meiosis rate.

NUMERICAL RESULTS

Benchmark problems

It is necessary for proposed algorithm to be tested on benchmark problem sets and compare it with other approaches. In this way the quality of proposed algorithm can be assessed using identical test problems. Each benchmark instances contains the array of processing time for n jobs and m machines ($P_{n \times m}$) which originally generated randomly. Each problem has two makespan references such as lower bound (LB) and upper bound (UB) values. The lower bound can be found by relaxing the capacity constraints on all that one machine [41]. The optimal is found for the simplified problem, where the solution is a lower bound of the original problem. The upper bound can be cut down by getting better solutions. The optimality can be reach if upper bound and lower bound get to the same point. Proposed algorithm has been tested on three different parts to find the robustness of algorithm.

As it shows in table 4, the results consist of three separate parts which the second part also breaks up in to two different parts.

In part 1, the problem sets with different sizes have been generated. Each problem has a known solution. Most of the problems have solved by using IP model to find the optimal solutions. The problems for the IP section have been solved in Cplex

12 on a Intel®Xeon® E 5-26650, RAM 64 GB of memory. The stopping criterions are either the relative optimality criteria (Optcr) reach to 0.05 when:

$$\frac{(|Obj - C_{best}|)}{(1.0-10+|C_{best}|)} < Optcr \qquad (12)$$

Here, *Obj* is the objective function value of the current integer problem and C_{best} is the best possible integer solution. Also the CPU time limit is set to 24 hours. All problems have been tested on NEH [2] and proposed GA (FCGA). The metric to check the performance of algorithm is relative percentage error [24] in makespan which formulated as:

$$\%\text{error} = 100 \cdot \frac{C_j^{nJmMk} - \min_{j=1,..,3} C_j^{nJmMk}}{\min_{j=1,..,3} C_j^{nJmMk}} \qquad (15)$$

Where the minimum makespan C_i^{nJmMk} of the benchmark instance nJmMk, n = 10,20,30,40,50,70,100;m = 5, 10, 15 , 20;k = 1,..,10 with respect to number of jobs, number of machines and the number of problem. To find the minimum makespan for each problem instance should use $\min_{j=1,..4} C^{nJmMk}$. For instance, for the first problem with the code 10J5M1 the generated solution is $C_1^{10J5M1} = 129$, IP solution $C_2^{10J5M1} = 121$, NEH result $C_3^{10J5M1} = 128$ and $C_4^{10J5M1} = 121$ for the FCGA best makespan over 10 runs. So, the minimum makespan for the first problem is $\min_{j=1,..4} C^{10J5M1} = 121$.

The second part of the testing is on 28 problem instances set up by Taillard [21] and 40 problem sets from Demirkol [23] studies. For the Taillard problem sets, FCGA compares with algorithms from Yagmahan and Yenise [24], Rajendran [25], Ravindran [26] and Rossi and Lanzetta [6]. Yagmahan and Yenise [23] proposed

multi-objective ant colony optimization (MOACSA) which combined ant colony optimization along with local search method to minimize total flow time and makespan for flow line scheduling problem. Rajendran algorithm starts by solving the problems with Campbell, Duck and Smith (CDS) algorithm [43] to find the best job sequence as a result of makespan minimization. Afterward, jobs are interchanged in the sequence to find the lowest makespan. Finally, this is used as the seed job sequence and possible improvement with respect to two objectives such as minimizing makespan and flow time are sought. The Ravindran [26] proposed hybrid algorithm which minimize makespan and total flow time by using the same idea as Rajendran [25] heuristic method.

For the Demirkol [23] benchmark problem sets ($part_{2,2}$) FCGA is compared with Yin and Lin [27], Lin and Yin [15] and Rossi and Lanzetta [6]. All algorithms in this part solved the problems by using non-permutation techniques and they tried to verify the effectiveness of their approach in comparison with other algorithms using a permutation solution. Yin and Lin [27] proposed ant colony optimization to solve the benchmark problems with the non-permutation method and proved that they obtained better solutions and judjed against other constructive heuristic methods with permutation technique. For hybrid simulated annealing and tabu search approach (SA-Tabu) , introduced by Lin and Yin [15], found outstanding results were for the Demirkol [23] benchmark problems.

Rossi and Lanzetta [6] proposed novel Ant colony optimization technique which used of heuristic design by the diagraph method to generate non-permutation flow shop schedules. They claimed the algorithm has high parallel capability for use

with modern processors. In this paper, all the problems instances from Taillard and Demirkol are tested with the Rossi and Lanzetta [6] algorithm and compared with FCGA. To measure the performance of algorithms two metrics from previous research [6] we used:

$$\%\, gap_{best} = 100 \cdot \frac{\sum_{j=1}^{9} C_{best}^{tai(10-j)} - \sum_{j=1}^{9} Best\ Known^{ta0i(10-j)}}{\sum_{j=1}^{9} Best\ Known^{ta0i(10-j)}}, \quad i = 0,1,2 \tag{16}$$

$$\%\, gap_{average} = 100 \cdot \frac{\sum_{j=1}^{9} C_{average}^{tai(10-j)} - \sum_{j=1}^{9} Best\ Known^{ta0i(10-j)}}{\sum_{j=1}^{9} Best\ Known^{ta0i(10-j)}}, \quad i = 0,1,2 \tag{17}$$

In equations 16, 17 ta0ij is the number of instance for i = 0,1,2 and j = 1,..,9 and $C_{best}^{tai(10-j)}$ is the minimum makespan for the specific problem by using specific algorithm. The metrics are going to find the relative distance between the upper bound and the best solution which is found by proposed algorithm.

The last part of the results are from tests on more complex problems which can be solved in reasonable amount of time using methods such as NEH [2] and Improved Heuristic method [13]. The results from two other heuristic presented and compared against the FCGA results. All the algorithms have been used the same system that it mentioned for IP model.

RESULTS

Part$_1$. As it shown in Table 5 this part consist of problems with job sizes of 10, 20, 30, 40, 50, 70, 100 and different machine sizes of 5, 10, 15, and 20. To have all range of problems the processing time generated random between the range of [1, 20] for the small problems and [1, 10] for the bigger problems. The results for problem

generator, IP model, NEH and FCGA are demonstrated in Table 5. The best result for each problem are shown in bold.

$Part_{2,1}$ and $Part_{2,2}$. In the first part, the four algorithms such as Yagmahan and Yenisey [24], Rajendran [25], Ravindran [26] and Rossi and Lanzetta [6] are tested on 28 standared test functions. Table 6 illustrates the final results for all problems.

$Part_{2,2}$ be composed of the results for four algorithms which they have been solved on the Demirkol [23] problems in a non-permutation ways versus the best and average of 10 runs values for FCGA algorithm. Table 7 illustrates the results of all approaches.

Table 8 represents the calculated %gap base on the formulations (16), (17) for the results in $part_{2,1}$ and $part_{2,2}$.

DISCUSSION

In $Part_1$, the two results from IP model and problem generator provides the robust benchmarks for the NEH and FCGA algorithms to compare with. It also assesses the quality of problem generator solutions and compared with optimal solutions.

The results show FCGA, except for three problems in 100×20 instances, has considerably better performance than the NEH. For the first three sets which consist of 10×5, 20×5 and 20×10 FCGA demonstrates less than 9% error in comparison with optimal solutions. However, FCGA encounter some difficulties with the problems size 40×15, where it has a 36% error. In a case of more complex problems, FCGA presents

the error less than 15% which still acceptable for the large size problems in a reasonable about of time.

The method for generating solutions for the flow line problems provides a robust benchmark to qualify the performance of other algorithms. Except for the 40×15 problem instances, problem generator itself yielded solutions less that 15% from optimality.

According to the results in table 8, consisting $Part_{2,1}$ and $Part_{2,2}$,FCGA algorithm shows superior results than the other algorithms which have been tested on Taillard and Demrikol datasets. It should be noted that NPFS ACO performs more efficient on the ta001 problem. According to the results (Table 8), FCGA has less than 3.2% gap for the Taillard problem instances. The proposed FCGA algorithm has shown competent performance on Demirkol benchmark with the low %gap of 1.79% for the small problems and the high %gap of 3.22% for the more complex problems.

It should be mentioned that Table 7 demonstrates a solid comparison between non-permutation and permutation techniques for solving problems. The proposed FCGA is the only algorithm which has been solved the problems with permutation techniques which outperform other four non-permutation algorithms.

The last part consists of the complex problem instances which can be solved by NEH [2] and Improved Heuristic [13] methods in a practical about of time. Except for one problem with the size of 100 × 100 in which NEH obtained better solutions, FCGA show better solutions than obtained with other two algorithms with the remaining problems.

The plots in figure 6 show the superior results of FCGA in comparison with other two algorithms for large size problems with higher complexity.

CONCLUSION

The inspirations of mechanisms in human genetics have been made proposed FCGA as an outstanding performance in compare with other heuristic algorithms for flow shop scheduling problems. Not only it has shown the best performance in permutation flow line configurations, it opposed other authors who applied computationally expensive non-permutation method for solving flow line problems.

Computational experiments have shown promising performance of such general purpose optimization tool regardless of problem complexity with large jobs and machine sizes.

ACKNOWLEDGMENT

This work was partially supported by University Of Rhode Island.

REFERENCES

1. Andrea Rossi , Alessio Puppato & Michele Lanzetta (2013) Heuristics for scheduling a two-stage hybrid flow shop with parallel batching machines: application at a hospital sterilisation plant, International Journal of Production Research, 51:8, 2363-2376
2. Nawaz, Muhammad, E. Emory Enscore, and Inyong Ham. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem." *Omega* 11.1 (1983): 91-95.
3. Potts, Chris N., David B. Shmoys, and David P. Williamson. "Permutation vs. non-permutation flow shop schedules." *Operations Research Letters* 10.5 (1991): 281-284.
4. Herroelen, Willy, Bert De Reyck, and Erik Demeulemeester. "Resource-constrained project scheduling: a survey of recent developments." *Computers & Operations Research* 25.4 (1998): 279-302.
5. French, S. "Sequencing and scheduling, mathematics and its applications." (1982).
6. Rossi, Andrea, and Michele Lanzetta. "Scheduling flow lines with buffers by ant colony digraph." *Expert Systems with Applications* 40.9 (2013): 3328-3340.
7. Fisher, Henry, and Gerald L. Thompson. "Probabilistic learning combinations of local job-shop scheduling rules." *Industrial scheduling* 3 (1963): 225-251.
8. Brah, Shaukat A., and John L. Hunsucker. "Branch and bound algorithm for the flow shop with multiple processors." *European Journal of Operational Research* 51.1 (1991): 88-99.

9. Johnson, Selmer Martin. "Optimal two-and three-stage production schedules with setup times included." *Naval research logistics quarterly* 1.1 (1954): 61-68.
10. Taillard, Eric. "Some efficient heuristic methods for the flow shop sequencing problem." *European journal of Operational research* 47.1 (1990): 65-74.
11. Ruiz, Rubén, and Concepción Maroto. "A comprehensive review and evaluation of permutation flowshop heuristics." *European Journal of Operational Research* 165.2 (2005): 479-494.
12. Pan, Quan-Ke, and Rubén Ruiz. "A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime." *Computers & Operations Research* 40.1 (2013): 117-128.
13. Chakraborty, Uday Kumar, and Dipak Laha. "An improved heuristic for permutation flowshop scheduling." *International Journal of Information and Communication Technology* 1.1 (2007): 89-97.
14. Nowicki, Eugeniusz, and Czeslaw Smutnicki. "A fast taboo search algorithm for the job shop problem." *Management science* 42.6 (1996): 797-813.
15. Lin, S-W., and K-C. Ying. "Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems." *International Journal of Production Research* 47.5 (2009): 1411-1424.
16. Li, B., Wu, S., Yang, J., Zhou, Y., & Du, M. (2012). A three-fold approach for job shop problems: A divide-and-integrate strategy with immune algorithm. *Journal of Manufacturing Systems*, *31*(2), 195-203.
17. Ruiz, Rubén, Concepción Maroto, and Javier Alcaraz. "Two new robust genetic algorithms for the flowshop scheduling problem." *Omega* 34.5 (2006): 461-476

18. Gen, M.; Tsujimura, Y.; Kubota, E., "Solving job-shop scheduling problems by genetic algorithm," in *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on* , vol.2, no., pp.1577-1582 vol.2, 2-5 Oct 1994.
19. Colin R. Reeves, A genetic algorithm for flowshop sequencing, Computers & Operations Research, Volume 22, Issue 1, January 1995, Pages 5-13, ISSN 0305-0548, http://dx.doi.org/10.1016/0305-0548(93)E0014-K.
20. Rubén Ruiz, Concepción Maroto, Javier Alcaraz, Two new robust genetic algorithms for the flowshop scheduling problem, Omega, Volume 34, Issue 5, October 2006, Pages 461-476, ISSN 0305-0483, http://dx.doi.org/10.1016/j.omega.2004.12.006.
21. Taillard, Eric. "Benchmarks for basic scheduling problems." *european journal of operational research* 64.2 (1993): 278-285.
22. Tandon, M., P. T. Cummings, and M. D. LeVan. "Flowshop sequencing with non-permutation schedules." *Computers & chemical engineering* 15.8 (1991): 601-607.
23. Demirkol, Ebru, Sanjay Mehta, and Reha Uzsoy. "Benchmarks for shop scheduling problems." *European Journal of Operational Research* 109.1 (1998): 137-141.
24. Yagmahan, Betul, and Mehmet Mutlu Yenisey. "A multi-objective ant colony system algorithm for flow shop scheduling problem." *Expert Systems with Applications* 37.2 (2010): 1361-1368.
25. Rajendran, Chandrasekharan. "Heuristics for scheduling in flowshop with multiple objectives." *European journal of operational research* 82.3 (1995): 540-555.

26. Ravindran, D., et al. "Flow shop scheduling with multiple objective of minimizing makespan and total flow time." *The international journal of advanced manufacturing technology* 25.9-10 (2005): 1007-1012.
27. Ying, Kuo-Ching, and Shih-Wei Lin. "Multi-heuristic desirability ant colony system heuristic for non-permutation flowshop scheduling problems." *The International Journal of Advanced Manufacturing Technology* 33.7-8 (2007): 793-802.
28. Bellman, Richard. "Mathematical aspects of scheduling theory." *Journal of the Society for Industrial and Applied Mathematics* 4.3 (1956): 168-205.
29. R. E. Smith and E. Smuda, “Adaptively resizing populations: Algorithm, analysis, and first results,” Complex Systems, vol. 9, no. 1, pp. 47–72, 1995.
30. T.-H. Li, C. B. Lucasius, and G. Kateman, “Optimization of calibration data with the dynamic genetic algorithm,” Analytica Chimica Acta, vol. 268, no. 1, pp. 123–134, 1992.
31. S. W. Mahfoud, “Niching Methods for Genetic Algorithms,” 1995: Illinois Genetic Algorithm Lab., Univ. Illinois.
32. J. C. Potts, T. D. Giddens, and S. B. Yadav, “The development and evaluation of an improved genetic algorithm based on migration and artificial selection,” IEEE Transactions on Systems, Man and Cybernetics, vol. 24, no. 1, pp. 73–86, 1994.
33. M. L. Mauldin, “Maintaining Diversity in Genetic Search.,” pp. 247–250, 1984.
34. J. Arabas, Z. Michalewicz, and J. Mulawka, “GAVaPS-a genetic algorithm with varying population size,” IEEE World Congress on Computational Intelligence, pp. 73–78 vol.1, 1994.

35. J. J. Grefenstette, “Optimization of Control Parameters for Genetic Algorithms,” IEEE Transactions on Systems, Man and Cybernetics, vol. 16, no. 1, pp. 122–128, 1986.
36. Herrera, Francisco, and Manuel Lozano. "Adaptation of genetic algorithm parameters based on fuzzy logic controllers." *Genetic Algorithms and Soft Computing* 8 (1996): 95-125.
37. M. A. Lee and H. Takagi, “Integrating design stages of fuzzy systems using genetic algorithms,” Proceeding 2nd IEEE International Conference. Fuzzy Systems, pp.612 -617, 1993.
38. B. H. Xu, R. C. Baird, and G. Vukovich, “Fuzzy Evolutionary Algorithms and Automatic Robot Trajectory Generation,” Methods and Applications of Intelligent Control, pp. 423–449, 1997.
39. H. Y. Xu and G. VUKOVICH, “A fuzzy genetic algorithm with effective search and optimization,” International Joint Conference on Neural Networks, vol. 3, pp. 2967–2970, 1993.
40. M. Lee and H. Takagi, “Dynamic control of genetic algorithms using fuzzy logic techniques,” Proceedings 5th International Conference genetic Algorithms, pp.76-83, 1993.
41. Pinedo, Michael. "Scheduling: theory, algorithms and systems, 1995."
42. D. H. Wolpert and W. G. Macready, “No fre lunch theorems for optimization”, IEEE Transactions on Evolutionary Computation, vol. 1, pp.67-82, 1997.

43. Campbell, Herbert G., Richard A. Dudek, and Milton L. Smith. "A heuristic algorithm for the n job, m machine sequencing problem." Management science 16.10 (1970): B-630.
44. Shirazi, Arash N., Meghan Steinhaus, and Manbir Sodhi.”Modified Genetic Algorithm Based on Human Cell Mechanisms”. Manuscript in preparation.

TABLES

Table 1. Parameters for proposed GA

Parameter	***Definition***	***Number***
m_1	Mitosis rate	[0 , 1]
m_2	Meiosis rate	[0 , 1]
$\alpha_1 . m_1$	ratio of chromosomes in N_1 that should place in N'_1	[0 , 1]
$\alpha_2 . m_2$	ratio of chromosomes in N_1 that should place in N'_2	[0 , 1]
$\beta_1 . m_1$	ratio of chromosomes in N_2 that should place in N'_1	[0 , 1]
$\beta_2 . m_2$	ratio of chromosomes in N_2 that should place in N'_2	[0 , 1]

Table 2. Fuzzy parameters in FCGA

Fuzzy parameters	Initiation (center 1)	processing (center 2)
Input 1	F_{best}/F_{ave}	F_{best}/F_{ave}
Input 2	F_{best}/F_{worst}	F_{best}/F_{worst}
Input 3	F_{change}	$(d_{ave} - d_{min})/(d_{max} - d_{min})$
$x_{1,1}$	0.3	$Final_{best}/Final_{ave}$
$x_{1,2}$	0.6	$1-Final_{best}/Final_{ave}$
$x_{2,1}$	0.3	$Final_{best}/Final_{worst}$
$x_{2,2}$	0.6	$1-Final_{best}/Final_{worst}$
$x_{3,1}$	0.3	0.3
$x_{3,2}$	0.6	0.6
$y_{1,1}$	0.5 × Popsize	0.5 × Popsize
$y_{1,2}$	0.8 × Popsize	0.8 × Popsize
$y_{2,1}$	0.5 × Popsize	0.5 × Popsize
$y_{2,2}$	0.8 × Popsize	0.8 × Popsize

Table 3. Fuzzy rules for meiosis procedure in FCGA

Rule	IF			THEN
Number	**Input fuzzy 1**	**Input fuzzy 2**	**Input fuzzy 3**	**Output fuzzy**
1	small	small	small	small
2	small	small	medium	medium
3	small	small	high	medium
4	small	medium	high	medium
5	small	high	small	medium
6	small	high	medium	medium
7	small	high	high	high
8	medium	small	small	medium
9	medium	small	medium	medium
10	medium	medium	medium	medium
11	medium	medium	high	high
12	medium	high	small	medium
13	high	small	medium	medium
14	high	small	high	high
15	high	medium	medium	high
16	high	medium	high	high
17	high	high	small	high

Table 4. Benchmark sets for the result parts

Parameters	Part $_1$	Part $_{2,1}$	Part $_{2,2}$	Part $_3$
# Jobs	10, 20, 30, 40, 50, 70, 100	20	20, 30, 40, 50	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
# Machine	5, 10, 15, 20	5, 10, 20	15, 20	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
Processing time	Random[1,10], Random[1,20]	Random[1,99]	Random[1,200]	Random[1,10], Random[1,50], Random[1,100]
n to m ratio	1-20	1-4	1-3.3	1
# Operations ($n \times m$)	50-2000	100-400	300-1000	100-10,000
# Instances per set	10	10	10	5
# Instances total	90	30	40	50
# Table for the result	Table	Table	Table	Table
competition approach	IP model, NEH[4],	Rajendran[25],Ravindran et al.[26], Yagmahan and Yenisey[24], Rossi and Lanzetto[6]	Demirkol et al.[23], Ying and Lin [27], Lin and Ying [15], Rossi and Lanzetta [6]	NEH[2], Improved Heuristic[13]
Solving method	Permutation	Permutation	Non-permutation	Permutation

Table 5. Performance measurement of algorithms base on error percentage

Algorithm Instance	Problem number	Solution generator	NEH	IP	FCGA
10 × 5 [1-20]	10J5M1	6.61	5.79	0.00	**0.00***
	10J5M2	0.00	7.62	0.00	**0.00***
	10J5M3	12.24	10.20	0.00	**0.00***
	10J5M4	0.00	18.37	0.00	**0.00***
	10J5M5	8.03	9.49	0.00	**1.46***
	10J5M6	0.00	10.43	0.00	**0.00***
	10J5M7	1.67	1.67	0.00	**0.00***
	10J5M8	8.57	21.90	0.00	**0.00***
	10J5M9	0.00	16.67	0.00	**0.00***
	10J5M10	2.91	12.79	0.00	**0.00***
20 × 5 [1-10]	20J5M1	10.48	7.62	0.00	**0.00***
	20J5M2	1.61	11.29	0.00	**0.00***
	20J5M3	1.36	8.16	0.00	**0.00***
	20J5M4	0.00	6.92	0.00	**2.31***
	20J5M5	4.55	5.19	0.00	**0.00***
	20J5M6	0.00	8.18	0.00	**0.91***
	20J5M7	7.92	7.92	0.00	**0.00***
	20J5M8	11.01	12.84	0.00	**1.83***
	20J5M9	0.85	6.78	0.00	**0.85***
	20J5M10	4.32	9.35	0.00	**0.72***
20 × 10 [1-10]	20J10M1	6.52	12.32	0.00	**7.25***
	20J10M2	0.00	19.51	0.00	**8.54***
	20J10M3	6.12	15.65	0.00	**4.76***
	20J10M4	2.58	14.19	0.00	**0.65***
	20J10M5	0.00	11.40	0.00	**4.15***
	20J10M6	4.55	8.44	0.00	**3.25***
	20J10M7	7.95	17.22	0.00	**6.62***
	20J10M8	0.00	10.16	0.00	**5.88***
	20J10M9	0.62	7.41	0.00	**5.56***
	20J10M10	2.56	11.54	0.00	**4.49***
20 × 20 [1-10]	20J20M1	0.00	11.16	0.00	**7.44***
	20J20M2	0.00	13.12	0.00	**11.31***
	20J20M3	0.00	18.64	0.00	**4.66***
	20J20M4	0.00	**6.76***	0.00	7.66
	20J20M5	2.07	10.36	0.00	**6.74***
	20J20M6	1.47	6.37	0.00	**0.49***
	20J20M7	1.05	18.85	0.00	**7.85***
	20J20M8	0.00	15.15	0.00	**12.12***
	20J20M9	3.40	11.06	0.00	**8.09***
	20J20M10	0.00	12.89	0.00	**8.00***

30 × 20 [1-10]	30J20M1	0.00	10.61	0.00	**8.71***
	30J20M2	0.00	13.74	0.00	**5.73***
	30J20M3	11.55	16.73	0.00	**8.37***
	30J20M4	0.00	18.15	0.00	**13.01***
	30J20M5	0.00	10.10	0.00	**6.62***
	30J20M6	0.00	12.82	0.00	**6.96***
	30J20M7	1.05	6.64	0.00	**0.00***
	30J20M8	3.86	16.60	0.00	**11.97***
	30J20M9	12.35	23.05	0.00	**17.70***
	30J20M10	4.65	13.18	0.00	**12.02***
40 × 15 [1-10]	40J15M1	0.00	7.85	0.00	**2.73***
	40J15M2	32.75	43.23	0.00	**35.81***
	40J15M3	15.05	31.77	0.00	**21.74***
	40J15M4	0.00	9.97	0.00	**7.90***
	40J15M5	0.00	6.77*	0.00	**6.77***
	40J15M6	0.00	16.32	0.00	**9.72***
	40J15M7	0.00	10.17	0.00	**6.78***
	40J15M8	6.41	11.54	0.00	**4.81***
	40J15M9	2.73	10.92	0.00	**8.87***
	40J15M10	2.47	19.08	0.00	**13.43***
50 × 20 [1-10]	50J20M1	1.65	11.54	0.00	**10.44***
	50J20M2	0.82	11.68	0.00	**8.97***
	50J20M3	1.02	13.74	0.00	**9.16***
	50J20M4	2.36	17.40	0.00	**13.86***
	50J20M5	2.70	10.24	0.00	**11.59***
	50J20M6	0.00	11.31	0.00	**5.91***
	50J20M7	0.00	13.24	0.00	**12.99***
	50J20M8	2.55	10.48	0.00	**5.95***
	50J20M9	0.00	16.96	0.00	**13.45***
	50J20M10	0.00	16.58	0.00	**13.90***
70 × 20 [1-10]	70J20M1	0.00	9.25	—	**7.71***
	70J20M2	0.00	9.27	—	**7.26***
	70J20M3	6.92	10.49	0.00	**7.59***
	70J20M4	0.38	11.86	0.00	**10.73***
	70J20M5	0.00	8.19	—	**7.33***
	70J20M6	0.00	9.48	—	**4.23***
	70J20M7	2.85	15.79	0.00	**14.47***
	70J20M8	0.00	11.30	0.00	**8.29***
	70J20M9	0.00	14.52	0.00	**13.91***
	70J20M10	0.00	14.67	—	**11.98***
100 × 20 [1-10]	100J20M1	0.00	10.88	—	**10.56***
	100J20M2	0.00	**10.97***	—	11.28
	100J20M3	0.00	9.20	—	**7.49***
	100J20M4	0.00	**5.41***	—	8.38
	100J20M5	0.00	**8.46***	—	9.64

	100J20M6	0.00	11.04	—	**10.73***
	100J20M7	0.00	12.20	—	**12.20***
	100J20M8	0.00	8.66	—	**8.35***
	100J20M9	0.00	9.89	—	**8.08***
	100J20M10	0.00	10.81	—	**10.81***

Table 6**.** Benchmark problems by Taillard [21], state of art solutions and computational results for Yagmahan and Yenisey [24], Rajendra [25], Ravindran [26], Rossi and Lanzetta [6] along with the results from proposed algorithm. The best prerformance has been shown in bold.

Algorithm Instance	#problem	UB/Best Known(Taillard's benchmark)	Yagmahan and Yenisey [24]	Rajendran [25]	Ravindran [26]	Rossi and Lanzetta [6] C_{best}	Rossi and Lanzetta [6] $C_{average}$	FCGA C_{best}	FCGA Caverage
20 × 5	ta001	1278	1297	1359	1297	**1290***	1293.1	1297	1297.5
	ta002	1358	1383	1378	1373	1389	1389	**1360***	1375
	ta003	1073	1203	1230	1206	1100	1112.5	**1098***	1114.4
	ta004	1292	1377	1393	1402	1344	1352.2	**1300***	1316.2
	ta005	1231	1311	1307	1334	1250	1258.7	**1243***	1254.9
	ta006	1193	1245	1282	1238	1217	1224.3	**1195***	1212.8
	ta007	1234	1303	1387	1322	1258	1259.6	**1251***	1253.4
	ta008	1199	1265	1344	1287	1235	1242.5	**1206***	1211.8
	ta009	1210	1303	1335	1307	1258	1275.3	**1255***	1267.9
	ta010	1103	1179	1191	1195	**1127***	1145.5	**1127***	1144.8
20 × 10	ta011	1560	1681	1711	1774	1693	1729.6	**1622***	1633.8
	ta012	1644	1749	1916	1791	1785	1799.5	**1700***	1719.9
	ta013	1486	1554	1617	1643	1583	1596.5	**1540***	1554.7
	ta014	1368	1490	1533	1531	1452	1478.3	**1401***	1433.9
	ta015	1413	1455	1588	1557	1516	1526.7	**1441***	1473.9
	ta016	1369	1564	1565	1612	1445	1468.3	**1417***	1438.3
	ta017	1428	1590	1622	1594	1524	1544.5	**1498***	1524.6
	ta018	1527	1595	1800	1631	1650	1663.8	**1555***	1611.7
	ta019	1586	1689	1717	1769	1659	1681.1	**1616***	1634.9

	ta020	1559	1719	1831	1744	1670	1677.7	**1624***	1641.4
20 × 20	ta021	2293	2428	2610	2491	2396	2414.9	**2326***	2354
	ta022	2092	2281	2301	2491	2225	2239.5	**2144***	2158.9
	ta023	2313	2515	2411	2422	2446	2464.5	**2375***	2395.2
	ta024	2223	2299	2471	2567	2346	2360.8	**2250***	2289.3
	ta025	2291	2473	2427	2420	2439	2460.5	**2318***	2356.6
	ta026	2221	2339	2466	2557	2331	2346.5	**2255***	2287.4
	ta027	2267	2378	2174	2448	2428	2454	**2316***	2344.1
	ta028	2183	2418	2418	2464	2321	2345.6	**2237***	2259.6

Table 7. Benchmark problems by Demirkol [23], state of art solutions and computational results for Lin and Ying [15], Demirkol [23], Ying and Lin [27], Rossi and Lanzetta [6] along with the results from proposed algorithm. The best prerformance has been shown in bold.

Algorithm Instance	# Problem	LB	Lin and Ying [15] C_{best}	Demirkol [23] C_{best}	Ying and Lin [27] C_{best}	Rossi and Lanzetta [6] C_{best}	Rossi and Lanzetta [6] $C_{average}$	FCGA $C_{average}$	FCGA C_{best}
20 × 15	flcmax-1 (20_15_3)	3354	3873	4437	4420	4047	4113.4	4034.9	**3955***
20 × 15	flcmax -2 (20_15_6)	3168	3761	4144	4044	3950	3977.5	3891.1	**3832***
20 × 15	flcmax-3 (20_15_4)	2997	3518	3779	3786	3692	3730.6	3649.5	**3589***
20 × 15	flcmax- 4 (20_15_10)	3420	4051	4302	4265	4176	4221.7	4173.9	**4119***
20 × 15	flcmax -5 (20_15_5)	3494	3913	4373	4310	4097	4124.9	4053.8	**3977***
20 × 20	flcmax-6 (20_20_1)	3776	4525	4821	4819	4790	4826.9	4680.3	**4583***
20 × 20	flcmax-7 (20_20_3)	3758	4435	4779	4723	4694	4715.7	4559.1	**4518***
20 × 20	flcmax -8 (20_20_9)	3902	4527	4944	4922	4720	4775.4	4655.7	**4598***
20 × 20	flcmax-9 (20_20_2)	3881	4499	4886	4847	4731	4781.3	4606.8	**4577***

20 × 20	flcmax-10 (20_20_10)	3823	4361	4717	4715	4554	4607.6	4513.2	**4470***
30 × 15	flcmax-11 (30_15_3)	4020	4568	5226	5210	4927	5032.2	4758.9	**4656***
30 × 15	flcmax-12 (30_15_4)	4080	4649	5304	5284	5033	5092.4	4846.6	**4802***
30 × 15	flcmax-13 (30_15_9)	4022	4568	5079	5075	4912	4968.6	4845	**4789***
30 × 15	flcmax-14 (30_15_8)	4490	4836	5605	5593	5220	5320.2	4725.6	**4660***
30 × 15	flcmax-15 (30_15_6)	4184	4761	5147	5149	5097	5158.5	5031.8	**4964***
30 × 20	flcmax-16 (30_20_3)	4806	5376	6183	5987	5794	5846.9	5545.6	**5472***
30 × 20	flcmax-17 (30_20_1)	4772	5698	6037	5989	6179	6221.8	5883.9	**5798***
30 × 20	flcmax-18 (30_20_6)	5004	5752	6241	6195	6039	6133.9	5916	**5868***
30 × 20	flcmax-19 (30_20_10)	4899	5464	6095	5923	5888	5967.7	5638.4	**5594***
30 × 20	flcmax-20 (30_20_2)	4757	5369	5822	5840	5842	5886.1	5565.2	**5491***
40 × 15	flcmax-21 (40_15_5)	5560	5958	6986	6972	6521	6594.1	6247.4	**6095***
40 × 15	flcmax-22 (40_15_9)	5119	5692	6351	6310	6244	6303.3	5963.2	**5877***
40 × 15	flcmax-23 (40_15_2)	5290	5877	6506	6532	6302	6395.6	6093.1	**6031***
40 × 15	flcmax-24	5596	5896	6845	6712	6413	6445.2	6107.4	**6035***

	(40_15_10)								
40 × 15	flcmax-25 (40_15_8)	5576	6054	6783	6771	6526	6611.7	6264.9	**6201***
40 × 20	flcmax-26 (40_20_3)	5693	6508	7154	7132	7208	7274.5	6855.1	**6738***
40 × 20	flcmax-27 (40_20_9)	5998	6676	7528	7496	7388	7484.7	6953.6	**6855***
40 × 20	flcmax-28 (40_20_6)	5990	6798	7469	7476	7455	7553.1	7123.5	**7024***
40 × 20	flcmax-29 (40_20_7)	6170	6766	7608	7588	7405	7473.5	7059	**6929***
40 × 20	flcmax-30 (40_20_5)	6011	6508	7219	7217	7326	7399.4	6863.6	**6715***
50 × 15	flcmax-31 (50_15_6)	6290	6836	7673	7631	7559	7606.8	7106.6	**7034***
50 × 15	flcmax-32 (50_15_5)	6355	6672	7679	7496	7317	7368.4	6914.6	**6832***
50 × 15	flcmax-33 (50_15_1)	6198	6580	7416	7402	7205	7303.8	6918.7	**6845***
50 × 15	flcmax-34 (50_15_8)	6312	6799	7548	7558	7348	7468.4	7138.6	**7042***
50 × 15	flcmax-35 (50_15_2)	6351	6954	7750	7712	7547	7644.8	7221.2	**7176***
50 × 20	flcmax-36 (50_20_2)	6740	7682	8838	8836	8436	8684.4	8098.6	**8044***
50 × 20	flcmax-37 (50_20_1)	6736	7313	8539	8521	8064	8189.7	7649.7	**7496***
50 × 20	flcmax-38 (50_20_7)	6756	7622	8417	8425	8370	8526	7930.5	**7825***

50 × 20	flcmax-39 (50_20_8)	6897	7480	8590	8536	8430	8509.2	7753.6	**7635***
50 × 20	flcmax-40 (50_20_4)	6830	7726	8493	8502	8538	8625.1	8117.2	**7999***

Table 8. Collection of results from Taillard and Demirkol benchmarks with the size of n × m. The Yagmahan and Yenisey [24] , Rajendran [25], and Ravindran [26] gap percentages are going to test on Taillard and Lin and Ying [15], Demirkol [23],and Ying and Lin [27] on Demirkol bechmark. Rossi and Lamzetta [6] and FCGA have the results for both algorithm. The best perfomed algorithm shown in bold.

Algorithm Instance	Benchmark	Yagmahan and Yenisey [24]	Lin and Ying [15]	Rajendran [25]	Ravindran [26]	Demirkol et al.[23]	Ying and lin [27]	Rossi and Lamzetta [6]	FCGA
20 × 5	Taillard	6.49	—	8.50	5.71	—	—	2.44	**1.32***
20 × 10	Taillard	11.42	—	13.12	7.67	—	—	6.94	**3.17***
20 × 15	Taillard	11.06	—	7.80	6.98	—	—	5.87	**1.89***
20 × 15	Demirkol	—	0.0	—	—	10.04	8.94	4.43	**1.86***
20 × 20	Demirkol	—	0.0	—	—	8.05	7.51	5.11	**1.79***
30 × 15	Demirkol	—	0.0	—	—	12.74	12.53	7.73	**2.09***
30 × 20	Demirkol	—	0.0	—	—	9.83	8.23	7.53	**2.04***
40 × 15	Demirkol	—	0.0	—	—	13.55	12.96	8.58	**2.59***
40 × 20	Demirkol	—	0.0	—	—	11.19	10.98	10.60	**3.02***
50 × 15	Demirkol	—	0.0	—	—	12.48	11.70	9.26	**3.22***
50 × 20	Demirkol	—	0.0	—	—	13.36	13.21	10.62	**3.11***

Table 9. Computational results for NEH[2], Improved Heuristic [13], and FCGA on large size problems.

Problem Instances	Range	# problem	NEH [2]	Improved Heuristic [13]	FCGA $C_{average}$ (10 runs)	FCGA C_{best}
10 × 10	[1,10]	1	102	98	96	**96***
		2	110	108	100.8	**100***
		3	121	117	108.6	**108***
		4	112	112	107.6	**107***
		5	116	113	105	**105***
20 × 20	[1,10]	1	235	236	224.2	**221***
		2	227	227	220.8	**219***
		3	239	229	226.4	**225***
		4	229	235	226	**224***
		5	238	243	226.2	**224***
30 × 30	[1,10]	1	362	359	351.6	**347***
		2	353	353	351.2	**350***
		3	365	360	352	**351***
		4	370	362	356.8	**354***
		5	372	363	358.6	**354***
40 × 40	[1,10]	1	484	487	478.2	**473***
		2	491	487	477.8	**473***
		3	479	473	463.8	**457***
		4	497	499	491	**487***
		5	490	483	475.2	**472***
50 × 50	[1,10]	1	609	612	606.2	**604***
		2	628	633	609.6	**603***
		3	610	610	609.4	**605***
		4	622	628	612	**609***
		5	612	610	604.6	**599***
60 × 60	[1,50]	1	3852	3860	3787.4	**3762***
		2	3846	3840	3779.8	**3757***
		3	3852	3830	3778.6	**3746***
		4	3797	3841	3804	**3772***
		5	3801	3801	3756	**3704***
70 × 70	[1,100]	1	8988	8867	8823	**8753***
		2	8909	8870	8850.4	**8778***
		3	9007	9016	8834	**8749***
		4	8983	8932	8793.4	**8770***
		5	9008	8981	8988.6	**8960***
80 × 80	[1,100]	1	10380	10373	10283.6	**10234***

		2	10335	10367	10223.2	**10205***
		3	10319	10339	10361	**10315***
		4	10248	10179	10178.4	**10084***
		5	10406	10233	10251.2	**10200***
90 × 90	[1,50]	1	5866	5897	5801.4	**5776***
		2	5813	5865	5856.6	**5803***
		3	5815	5868	5818	**5777***
		4	5789	5791	5775.4	**5746***
		5	5804	5745	5781	**5744***
100 × 100	[1,10]	1	1279	1274	1273.4	**1262***
		2	**1264***	1269	1270.2	**1264***
		3	1272	1274	1274.8	**1267***
		4	1280	1271	1273	**1258***
		5	**1264***	1272	1275.4	1269

FIGURES

Figure 1. Permutation versus non-permutation

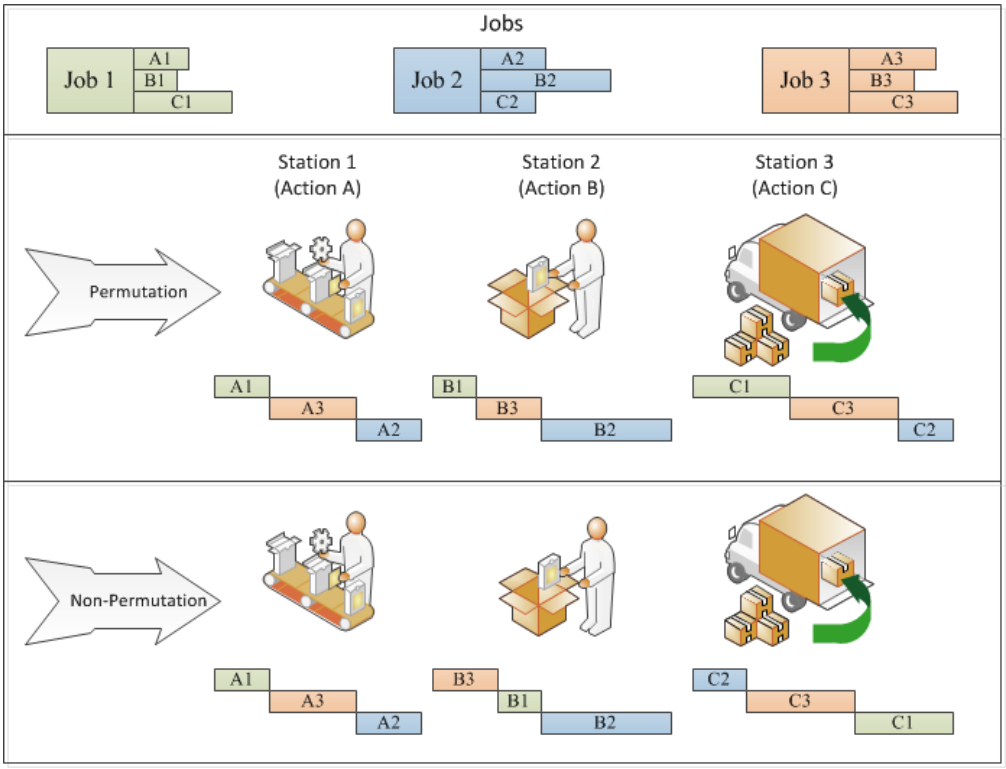


Figure 2. Proposed GA flowchart

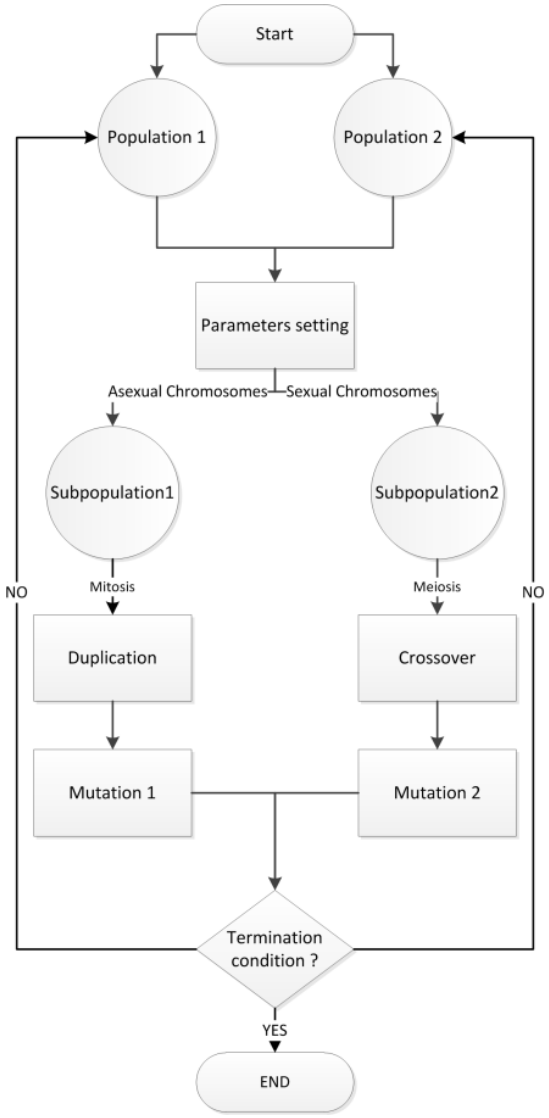


Figure 3. General form of using fuzzy technique to control GA parameters

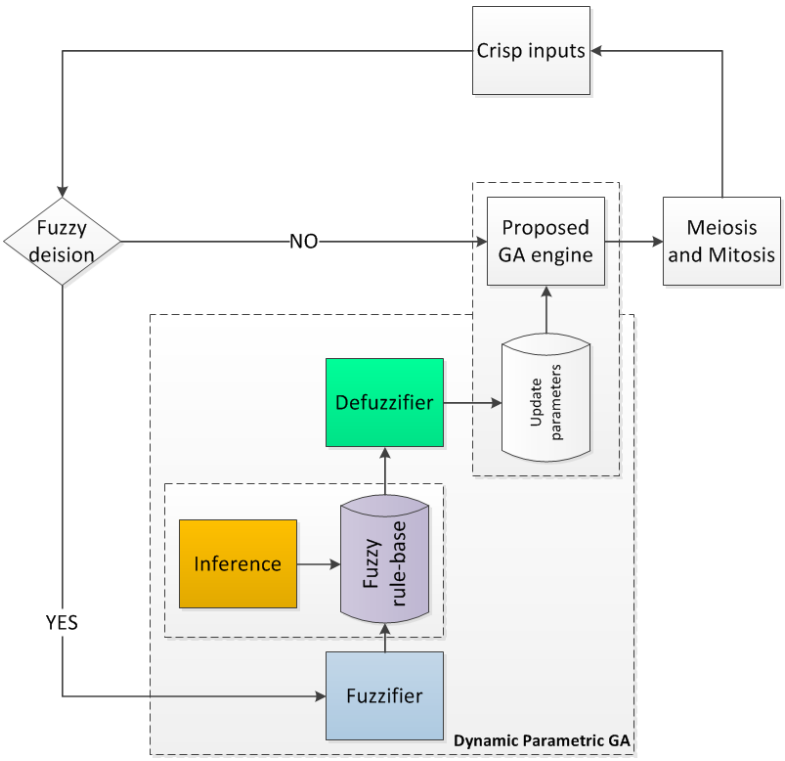


Figure 4. Example of triangular membership function with two centers (A, B)

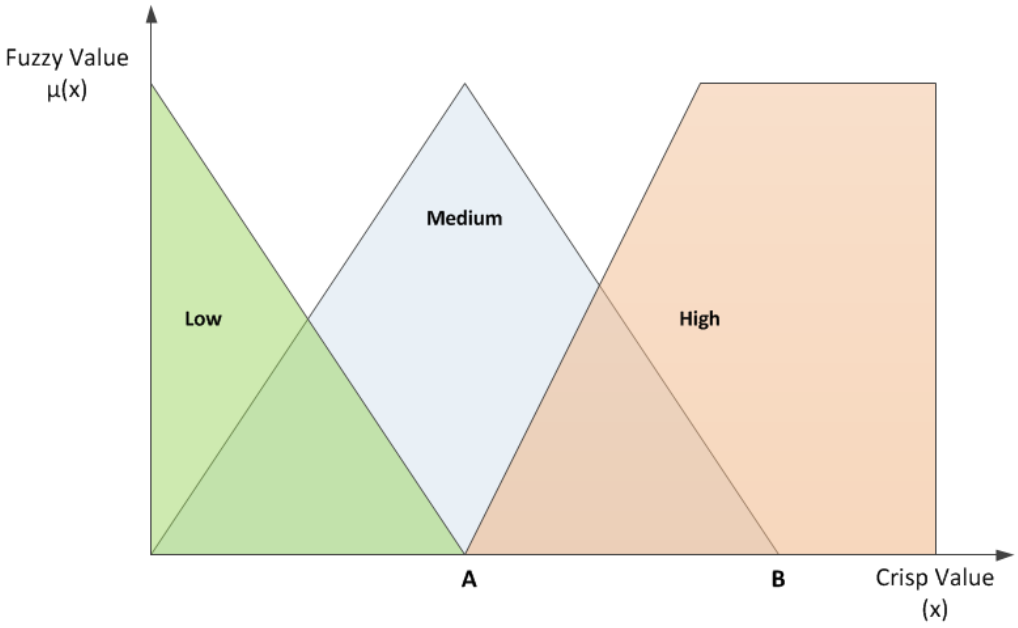


Figure 5. Converting crisp input data in to input fuzzy by using triangular membership function

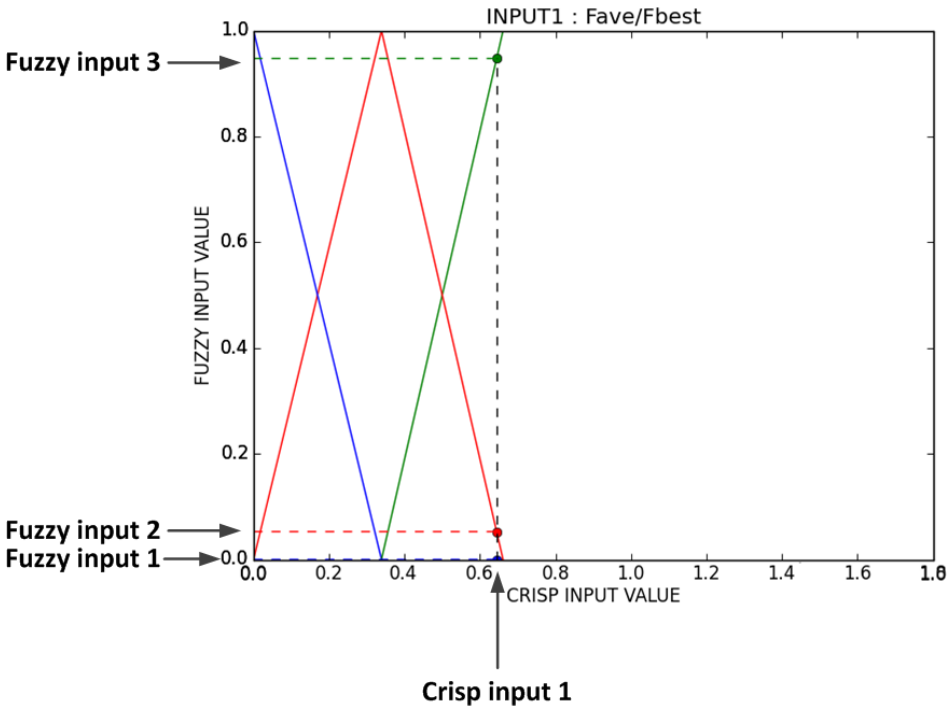
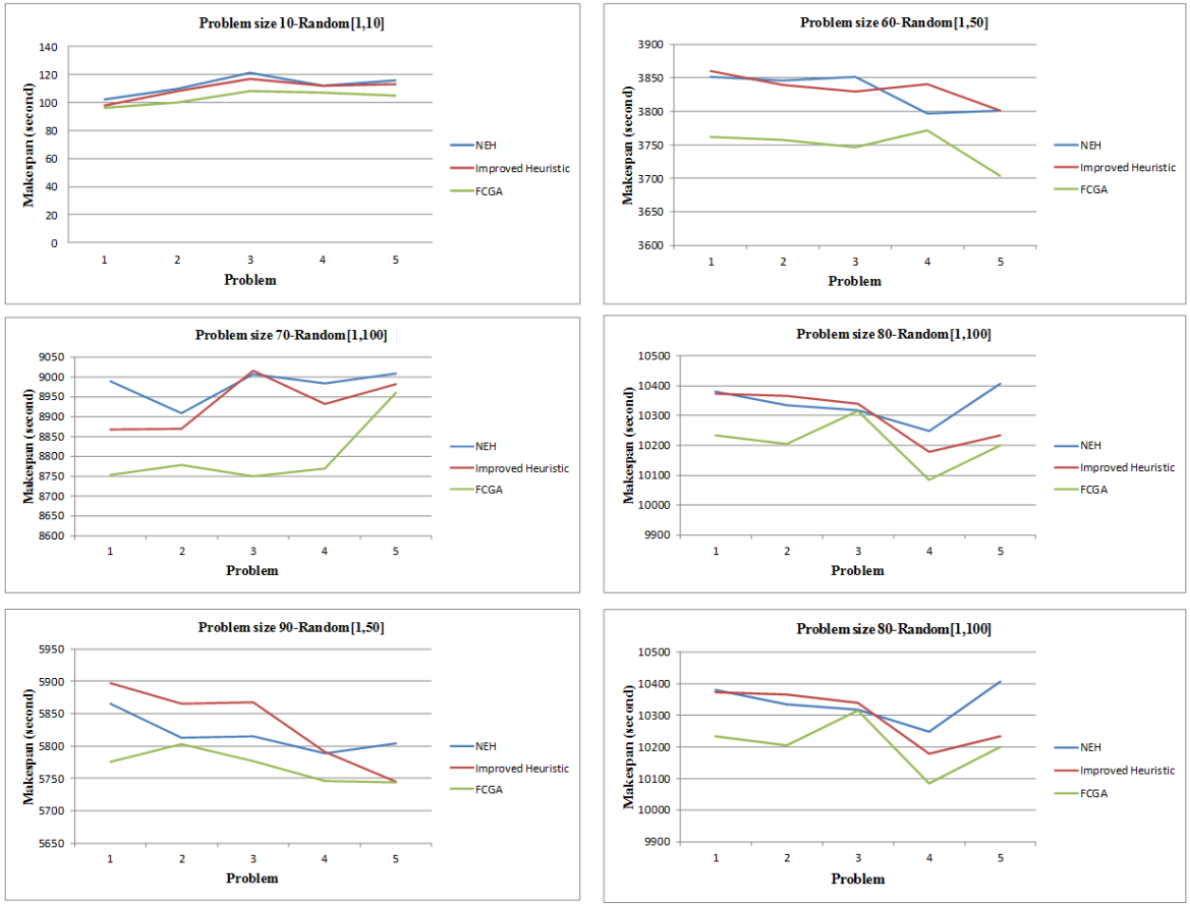


Figure 6. Comparison of FCGA, NEH and Improved Heuristic in large problem instances with n × m when n = m. In here just presented 6 problems sets such as 10×10, 60×60, 70×70, 80×80, 90×90 and 100×100 as the examples.



MANUSCRIPT – III: FUZZY CELL GENETIC ALGORITHM APPROACH FOR FLEXIBLE FLOW LINE SEQUENCING MODEL

In preparation for submission to International Journal of Operation Research

Arash Nasrolahi Shirazi[1], Meghan Steinhaus[1], Matthew Agostinelli[1], and Manbir Sodhi[1]

[1]Department of Mechanical, Industrial and Systems Engineering, University of Rhode Island, Kingston RI 02881

Corresponding author: Arash Nasrolahi Shirazi

Department of Mechanical,

Industrial &Systems Engineering

University of Rhode Island

Kingston, RI 02881

E-Mail: arashshirazi@uri.edu

ABSTRACT

This paper focuses on makespan minimization for the flexible flow line sequencing problem using a Fuzzy Cell Genetic Algorithm (FCGA). Real world applications of this problem can commonly be found in printing and electronic circuit board manufacturing industries. A generalized integer programming (IP) model for this problem is proposed. The Fuzzy Cell Genetic Algorithm (FCGA) is proposed to solve the IP model, which has been proven to be NP-hard. Sample problems are generated with known good solutions to evaluate the effectiveness of the FCGA approach. The FCGA matches the performance of the IP model for small sized problem instances and it is proven to be effective for larger problem instances.

Keywords: flow line sequencing, integer programming, fuzzy cell genetic algorithm

INTRODUCTION

The flow line sequencing problem is defined by a set of tasks, each task has priority as they are performed from one machine to another one in a manufacturing processes. The sequence determines the position of each task related to other jobs in individual machine. The general form has been taken by traditional manufacturing systems. The expanded version of flow line sequencing is Flexible flow line problem which provides more than one machine in parallel for all operations to provide greater availability for jobs to process.

The applications of flexible flow line occur in many real world environments including circuit board manufacturing, automobile manufacturing, paper, textile, and also production of concrete [1-6]. The flexible flow line systems can be defined as a set of n jobs are to be performed in m stages with p number of parallel machines to optimizing a given objective function. Generally, flexible flow line problems have follow features in common:

1. The number of stages to perform the jobs has to be more than one.
2. The total number of stages has to be less than the total number of machines.
3. All jobs have to be performed in all stages but it is optional for them to choose which machine to be processed on.
4. Each job has a processing time which represented by $Pt_{i,j}$ where i is the i^{th} job to perform and j is the j^{th} number of stage.

In general form of flexible flow shop problem, all machines in each stage are ready to perform the jobs at time zero, the parallel machines in each stage are

identical, each machine can perform just one job at a time and any job can be processed by any machine at each stage. Set up time is negligible in general and there is no limit for the buffers between the stages. Figure 1 represents the general form of problem design. The jobs signify by j, the stages define by s and each machine shows by p in each work center to perform the actions.

The flexible flow line problems are categorizing in to two kinds based on their performance [7]. The total flexibility which is more close to the general form is all operations are reachable on all available machines. The second model is partial flexibility where some operations are only doable on some the accessible machines. Previously, the researchers have been proved the Flexible flow line machine as a NP-hard class of problem and solutions should be approached by heuristic and metaheuristic methods for near-optimal solution [8], [9].

In the general form of the flexible flow line, the main objective is minimizing makespan as the scheduling criteria. To minimize the makespan should reduce the maximum completion time for all jobs. It means each job has a completion time in the last stage and the maximum of all completion times is the makespan for the system. Traditionally, makespan has been considered as an objective for much of the flexible flow line literature. But some other researchers used the maximum capacity volume and flow time as the objective [10].

In this paper, the flexible flow shop method with makespan objective is formulated by an integer programming (IP) model and also it's solution times and

quality are compared with the solutions obtained using fuzzy cell genetic Algorithm (FCGA) for the large size problems.

LITERATURE REVIEW

This literature review focuses on publications pertaining to scheduling in a flexible flow shop. Previously introduced approaches are discussed in regard to their advancement of research in the flexible flow shop scheduling problem as well as their formulation method. Approaches can generally be categorized as exact method algorithms, heuristic approaches, and metaheuristic algorithms.

Exact methods

Branch and bound algorithms have been proved to be the preferred technique in solving flow line sequencing problems to optimality. The majority of past research implementing branch and bound techniques has been focused on simplified flexible flow shops for which only two stages of production exist. Typically, only one of these stages will feature identical machines working in parallel while the other will feature a single machine.

The first set up of branch and bound being used on a general model of the flexible flow shop was introduced by Brah and Hunsucker [11]. They included the complex lower bound calculations to narrow the scope of the problem while minimizing makespan. The proposed branch and bound method was suitable only for problem instances of limited size. Rajendran and Chaudhuri [12] also proposed a branch and bound method for which they minimized makespan and restricted solutions

to the set of permutation schedules. Moursli and Pochet [13] adopted a branch and bound method similar to previous research but began using heuristics to construct numerous upper and lower bounds. The result of their research improved computational efficiency thus increasing suitability for incrementally larger and more difficult problems.

Santos et al.[14] used similar methods of permutation scheduling by only considering the permutations through the first stage and using first in first out (FIFO) queuing at every stage thereafter. More specific variants of the generalized flow shop have also been researched in which sequence dependent setup times and other production factors are considered.

Fattahi et al.[15] proposed a branch and bound algorithm for a flexible flow shop for which setup time and assembly time are both considered. They implemented new upper and lower bounds to constrain the solution set between concurrently solved stages in order to increase the efficiency of previously implemented branch and bound algorithms.

Heuristic

The NP-completeness of the flexible flow shop problem, proven by Gupta [16], confirms the need and motivation to present heuristic approaches. In Gupta et al. [17] research, a new lower bound was proposed for branch and bound as well as two constructive heuristics based on Johnson's Rule. Each was applied specifically to the case of two parallel machines in the first stage followed by a single machine in the second. Similarly, Sriskandarajah and Sethi [18] examined the best and worst case

performance of heuristic algorithms in a simple two stage flexible flow shop based on Johnson's Rule. Furthermore, Brah and Loo [19] evaluated the effectiveness of various heuristic methods that had not been applied to flow shops with multiple processors. They conducted their comparison on problem sets with varying number of jobs, stages, and machines and found the previously developed heuristics performed equitably with parallel machines.

Linn and Zhang [20] surveyed flexible flow shop scheduling and found heuristics developed by Kochhar and Morris [21] for k-stage parallel machines where blocking and setup times were considered among other production factors. They concluded that the heuristics performed close enough to optimality for practical use.

Metaheuristics

The application of metaheuristics which exploit heuristic methods repeatedly have proven to be effective, particularly in scheduling of flexible flow shops with multiple parallel stages and increasingly large numbers of jobs. The majority of metaheuristics researched focus on permutation scheduling in which a single order of jobs is applicable to all stages.

Ruiz and Vazquez-Rodriguez [22] found that initial approaches for the flexible flow shop focus on Tabu Search algorithms, such as those presented by Voss [23] and Haouari and M'Hallah [24]. Nowicki and Smutnicki [25] extended their previous work on flow shop scheduling, Nowicki and Smutnicki [26], proposed an improvement algorithm with focused on specific neighborhoods and employs notions of a critical path in a graph and blocks of jobs. Numerous other authors have

researched specific variants of the hybrid flow shop using Tabu Search, including limited buffers, no predecessor relationships, and blocking. Chen et al. [27] applied Tabu Search in the real world application of loading container ships and emphasized the importance of a strong initial solution to increase the efficiency of Tabu Search.

Simulated Annealing algorithms have also proven popular in optimization of the flexible flow shop. Naderi et al. [28] presented an improved simulated annealing for a flexible flow shop with sequence dependent setup times and transportation times between stages. A performance comparison only considering sequence dependent setup times proved the effectiveness of simulated annealing against other metaheuristics, including another simmulated annealing algorithm proposed by Jin et al. [29], a random keys genetic algorithm proposed by Kurz and Askin [30], and an artificial immune algorithm proposed by Zandeih et al. [31]. Mirsanei et al. [32] conducted a similar comparison considering sequence dependent setup times on the same previously proposed methods and achieved similar results that bolstered the effectiveness of simulated annealing algorithms.

In recent publications, numerous genetic algorithms have been proposed for a multitude of scheduling problem variations. The proposed random keys genetic algorithm of Kurz and Askin [30] was applied to a flexible flow shop with sequence dependent setup times.

Ruiz and Maroto [33] also employed a genetic algorithm for scheduling of jobs in a flexible flow shop in which machine eligibility and sequence dependent setup times are considered. They expanded on previous GA implementations and

introduced new crossover operators. Testing on generated problem sets and data from the ceramic tile industry, their proposed GA produced superior solutions in comparable CPU time to other genetic algorithms. Oguz and Ercan [34] proposed four version of GAs with novel crossover operations and compare the best one with Tabu search in a case of final solution and running time. The final results proved proposed GA had performed superior than Tabu search. Later on, Engin et al. [35] developed novel GA with modified mutation operation and compare their results with Qguz and Ercan [34] study to measure the performance of the algorithm. They found the proposed GA is very effective in terms of minimizing the makespan on problems.

FLEXIBLE FLOW LINE SCHEDULING PROBLEM DESCRIPTION

The mathematical formulation for the flexible flow line for the two-stage flexible flow shop with the objective of minimizing the makespan was presented by Guirchoun et.al [36]. Kurz and Askin [31] proposed the mathematical model for flexible flow line with minimizing the makespan by considering the sequence-dependent setup time.

The integer program formulation which addresses the problem in this paper will be shown here. Let t be the number of tasks to be schedule and i number of parallel machines at station s. The N, W and P are representing sets of jobs, stages and machines in parallel respectively. The definitions for this problem presented as:

Indices

t index of job to be schedule $(t = 1,2,\ldots,n)$

s index of station *(s = 1,2,…,m)*

k index of machine in each station *(k = 1,2,…,p)*

mn last station

Parameters

$Pt_{t,s}$ processing time of job t in station s

$Ct_{s,k}$ completion time of all jobs in station s and machine k

$Ct_{mn,k}$ completion time of all jobs in the last station mn and machine k

$Ft_{t,s,k}$ finishing time of task t in machine k in station s

Decision variable

$x_{0\ t,k}$ 1 if job t performs as a first job in parallel machine k and 0 otherwise

$x_{n\ t,k}$ 1 if job t performs as a last job in parallel machine k and 0 otherwise

$x_{i,j,k}$ 1 if job i performs before job j in parallel machine k and 0 otherwise

$$\text{Objective: min z,} \tag{13}$$

s.t.

$$\sum_{t\in N} x_{0\ t,k} = 1, \qquad \forall\, k \in P \tag{14}$$

$$\sum_{k\in P} x_{0\ t,p} + \sum_{k\in P}\sum_{j\in N} x_{t,j,k} = 1, \qquad \{\forall\, t,j \in N |\ t \neq j\} \tag{15}$$

$$x_{0\ t,k} + \sum_{i\in N} x_{i,t,k} = x_{n\ t,k} + \sum_{j\in N} x_{t,j,k}\ , \qquad \forall\, k \in P, \{\forall\, i,j,t \in N | i,j \neq t\} \tag{16}$$

$$Ct_{s,k} - Ft_{t,s,k} \geq 0\ , \qquad \forall s \in W\ , \forall\, k \in P, \forall\, t \in N \tag{17}$$

$$Ft_{t,1,k} - \sum_{i\in N} Pt_{i,1} x_{0\ i,k} \geq 0, \qquad \forall k \in P \tag{18}$$

$$Ft_{t,s,k} - Ft_{t,s-1,k} - \sum_{i\in N} Pt_{i,1} x_{0\ i,k} \geq 0, \qquad \{\forall s \in W | s > 1\}, \forall\, k \in P, \forall\, t \in N \tag{19}$$

$$Ft_{j,1,k} - Ft_{i,1,k} \geq Pt_{j,1} x_{i,j,k} - Pt_{i,1} x_{j,i,k} - \qquad \{\forall\ \mathrm{i,j} \in \mathrm{N} |\ \mathrm{i} \neq \mathrm{j}\} \tag{20}$$

$$(M - Pt_{j,1})1 - x_{i,j,k} - x_{j,i,k},$$

$$Ft_{j,s,k} - Ft_{i,s,k} \geq Pt_{j,s} x_{i,j,k} - Pt_{i,s} x_{j,i,k} - \{\forall s \in W \mid s > 1\}, \{\forall\, i,j \in N |\ i \neq j\}, \tag{21}$$

$$(M - Pt_{j,s})1 - x_{i,j,k} - x_{j,i,k}, \qquad \forall\, k \in P$$

$$Ft_{j,s,k} - Ft_{j,s-1,k} \geq Pt_{j,s}x_{i,j,k}, \quad \{\forall s \in W \mid s > 1\}, \{\forall \text{ i,j} \in \text{N} \mid \text{i} \neq \text{j}\} \tag{22}$$

$$z \geq Ct_{mn,k}, \quad \forall k \in P \tag{23}$$

$$x_{i,j,k} = 0, \quad \forall k \in P, \{\forall \text{ i,j} \in \text{N} \mid \text{i} \neq \text{j}\} \tag{24}$$

$$x_{0\,t,k} \in \{0,1\}, \quad \forall k \in P, \text{t} \in \text{N} \tag{13}$$

$$x_{L\,t,k} \in \{0,1\}, \quad \forall k \in P, \text{t} \in \text{N} \tag{14}$$

$$x_{i,j,k} = 0, \quad \forall k \in P, \{\forall \text{ i,j} \in \text{N} \mid \text{i} = \text{j}\} \tag{15}$$

We assume each task in each station can be performs at the same processing time in each parallel machine. Also, all tasks in the current station must be done before moving to the next station. The sequences of the jobs are the same from the first station to the last station. There is a restriction in this model that every stage must be visited by at least as many jobs as there are machines in that stage.

Eq.(1) represents the objective function which is to minimize the makespan z. Eq.(2) defines the constrain to assign at least one job as a starting job to each machine in parallel for each station. Constrain (3) ensure that job t is either the first job or should be done immediately after job j in machine k. Eq.(4) represents all the possibilities for each job in each machine to perform. Constrain set (5) makes sure that the completion time of all the jobs in machine k and station s should be greater than or equal to the finishing time of each individual task in same machine and station. In Eq.(6), finishing time of the first job which assigned to the machine k at the first station should be greater than or equal to the processing time of the first job at the same machine. Constrain set (7) enforce sequencing job t in each station s greater than 1 by putting the finishing time of each job in station s greater than the finishing time of the same task at the station before. Eq.(8) is going to calculate the accumulation of

processing time for the set of tasks in each machine at the first station. Constrain set (9) and (10) calculate the finishing time of job t as:

If t is the first job in parallel *k* at station *s*:

$$Ft_{t,s,k} = Ft_{t,s-1,k} + Pt_{t,s} \qquad \{\forall s \in W | s > 1\}, \forall k \in P, \forall t \in T \tag{25}$$

Else if t is the successor of the task j in parallel l at station s:

$$Ft_{t,s,k} = \max\left(Ft_{t,s-1,k}, Ft_{j,s,k}\right) + Pt_{t,s} \quad \{\forall t, j \in T \,|t \neq j\}, \{\forall s \in W | s > 1\}, \forall k \in P \tag{26}$$

Constrain set (11) links the decision variable z and $Ct_{mn,l}$ to minimize the completion time of the maximum machine in parallel at the last station mn. The last sets of equations (12),(13), (14), and (15) are defining the binary decision variables for the predecessor and successor job relations, first job, last job and zero for the case when the predecessor and successor have the same syntax in the problem.

FUZZY CELL GENETIC ALGORITHM

The modified genetic algorithm has been introduced in Shirazi, Steinhause, and Sodhi [37] which applied the main concepts of Meiosis and Mitosis in human cell division. Later on, Shirazi, Steinhause, and Sodhi [38] proposed Fuzzy Cell Genetic Algorithm (FCGA). They applied fuzzy logic to control the parameters in GA with human cell mechanism. It prevents the premature convergence and gets closer result to the optimality. The FCGA has shown superior performance in compare with other heuristic and metaheuristic algorithms for the flow line sequencing problems with the single machine in multiple stages with many jobs. In this paper, it is going to test FCGA algorithm on flexible flow shop problems. The FCGA has been presented in detail [38].

The FCGA starts with the random population of job sequences and the number of stages that they have to perform in each stage. The population is going to divide in to two sub-populations for the mitosis and meiosis procedures. All chromosomes will be evaluated by fitness function which represents the makespan for that particular job sequence. The Individuals in meiosis sub-population are going through the crossover and mutation procedures and the chromosomes in mitosis are going through duplication and small probability of mutation.

There are three metrics to measuring the FCGA performance and prevent it from premature convergence. F_{ave} defines as the mean of fitness distribution in the population and the F_{best} (best fitness) is measuring the best individual (lowest fitness value for minimization problems). The worst fitness (F_{worst}) is defined as the worst individual fitness value in the population. To calculate the changing rate (F_{change}) in a population, should calculate F_{worst} - F_{best} in a predefined number of iterations. It should be mention that the fitness value is the makespan for each sequence inside the chromosome. The three metrics are using as the inputs for fuzzy controller. The controller part is going to check the performance of the meiosis process. If the changing rate inside the sub-population decreases, the controller will start to increase the sub-population for mitosis to escalate solutions diversity and get away from premature convergence. However, the high changing rate will decrease the size of mitosis sub-population to not to diverge from optimum solution. Figure 2 shows the flow chart for FCGA. Asexual and sexual chromosomes are the group of chromosomes which should perform in mitosis and meiosis processes respectively.

GENERATING PROBLEM SETS

It is important to assess the performance of the FCGA on complex flexible flow line problems which IP model cannot solve them in an efficient period of time. Kruz and Askin [10] have been proposed the data generation for flexible flow shop by considering the setup times. However, they didn't clearly explain their method for generating the data sets.

In here, it is going to explain about the generating data sets in details and also provide an example for better understanding.

The requirement for generating data is n (number of jobs), s (number of stages), m (number of machines in parallel in each stage), the range of processing time for every i^{th} job in each j^{th} stage which defines by $Pt_{i,j}$ and $T_{j,k}$ is the sets of job in j^{th} stage and k^{th} parallel machine. It assumed the numbers of machines in each stage are the same and has to be more than one. The main purpose is to provide the sequence of jobs with no gap between the processes. The pseudocode of the problem generator is shown in Table1.

Figure 3 shows an example of 4 jobs, 2 stages and 2 machines in each stage. As it shown in the example, the final output for the problem generator is $PT_{4\times2} = \begin{bmatrix} 4 & 6 \\ 7 & 6 \\ 6 & 4 \\ 8 & 4 \end{bmatrix}$ with a solution of 18 sec.

It should be mention that the solution is not an optimal solution. However, it is a known solution which has no gap between the procedures. Obviously, the large size problems should be difficult for algorithms to solve.

The process time for each job generated random between the range of [1,10]. There are 3 sets of problems .The first set of problems consider number of jobs (n) = 5, 8, 10, 20, 30, 40, 50, 70, 100 and the number of stages (s) = 2, 5, 15 by assuming two machines in each stage. Next, set of 20, 30, 40, 50 and 100 jobs and stages along with 5 machines in each stage. Last data sets includes n = 20, 40, 100 and s = 10, 20, 40, 60, 100 with 10 machines in each stage. Table 2 represents the parameters for the data sets.

RESULTS AND DISCUSSION

Problems of small size have and can be solved by using IP model to find the optimal solutions. The problems for the IP section have been solved in Cplex 12 on Intel®Xeon® E 5-26650, RAM 64 GB of memory. The stopping criterions are either the relative optimality criteria (Optcr) reach to 0.05 when:

$$\frac{(|Obj - C_{best}|)}{(1.0-10+|C_{best}|)} < Optcr \qquad (27)$$

In here, Obj is the objective function value of the current integer problem and C_{best} is the best possible integer solution. A second limit is on the CPU time of 24 hours.

The IP model could only solve the problems with 5×2, 8×5, and 10×5 along with 2 machines in parallel. FCGA could find the optimal solutions for all problems

except for two problems with size 10×5 which have been solved by GAMS [39]. Figure 4 illustrate the line graph for the results of FCGA, and the IP model.

As it shown in figure 5, except for 11 problems over 100, FCGA could find superior results in compare with the problem generator in a case of 2 machines in each stage. The FCGA has shown outstanding searching performance in a solution space for the small size problems. However, problem generator hasn't shown great performance for the problem sets compared with the IP model. Therefore, it does not seem to be a robust reference point for the problems of larger sizes.

The second part which considers 5 machines in each stage, FCGA could get better results than the problem generator for just 16 problems over 50 total data sets (Figure 5). To compare the FCGA with generator solution, in this part, a "loss" measurement scale, where loss = (makespan - lower bound)/lower bound. Loss is accumulated for each subset of problems to asses FCGA performance.

Accumulated loss for problem instances 30×30, 40×40 and 50 × 50 are 0.27, 0.52, and 0.42 respectively. The highest total loss value for FCGA is 0.86 for 100×100 problem instances. On a per problem basis this represents a small amount of loss. The problem generator solutions have shown to be an efficient reference point for large size problems.

The last part is concentrated on more complex problems featuring by 10 machines in each stage which is difficult to find in other literatures. As it shown in Figure 6, FCGA performs much better for the problems with the size of [20×20], [40×40], [100×10], and [100×20] which the highest total loss is less than 0.2 for all

problems. However, the loss value increased by 0.77 for the problem set with size of [100×60]. For really large size problems [100×100] the loss goes up to 0.77 which is still small value for the FCGA performance on that complexity.

The problem generator could provide the superior solutions for the truly large size problem and provide robust references to assess the FCGA performance.

CONCLUSION

To find an efficient solution for flexible flow line problem is important as a consequence of implementing those solutions to the real life problems. The complexity of the problems makes them closer to real life situations. To formulate the IP model for the problems is valuable but it is not possible to solve the real size problems in a reasonable time. The process of generate the problem with the no gap between the processes is not going to give the optimalsolutions. However, it is fair enough to find the solutions better than that or even close to them for the large size problems. FCGA has been shown outstanding performance in other studies and also in this study for flexible flow line problems. Computational experiments have shown promising performance of this optimization approach regardless of problem complexity.

ACKNOWLEDGMENT

This work was supported partially by University Of Rhode Island.

REFERENCES

1 Agnetis, A., et al. "Scheduling of flexible flow lines in an automobile assembly plant." *European Journal of Operational Research* 97.2 (1997): 348-362.

2 Piramuthu, Selwyn, Narayan Raman, and Michael J. Shaw. "Learning-based scheduling in a flexible manufacturing flow line." *Engineering Management, IEEE Transactions on* 41.2 (1994): 172-182.

3 Aghezzaf, El-Houssaine, and Hendrik Van Landeghem. "An integrated model for inventory and production planning in a two-stage hybrid production system." *International journal of production research* 40.17 (2002): 4323-4339.

4 Grabowski, Jozef, and Jaroslaw Pempera. "Sequencing of jobs in some production system." *European Journal of Operational Research* 125.3 (2000): 535-550.

5 Jin, Z. H., et al. "SCHEDULING HYBRID FLOWSHOPS IN PRINTED CIRCUIT BOARD ASSEMBLY LINES*." *Production and Operations Management* 11.2 (2002): 216-230.

6 SHERALI, HANIF D., SUBHASH C. SARIN, and MURALIDHARAN S. KODIALAM. "Models and algorithms for a two-stage production process." *Production Planning & Control* 1.1 (1990): 27-39.

7 Kacem, Imed, Slim Hammadi, and Pierre Borne. "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 32.1 (2002): 1-13.

8 Gupta, Jatinder ND. "Two-stage, hybrid flowshop scheduling problem." *Journal of the Operational Research Society* (1988): 359-364.

9 Garey, Michael R., and David S. Johnson. "A Guide to the Theory of NP-Completeness." *WH Freemann, New York* (1979).

10 Kurz, Mary E., and Ronald G. Askin. "Comparing scheduling rules for flexible flow lines." *International Journal of Production Economics* 85.3 (2003): 371-388.

11 Brah, Shaukat A., and Luan Luan Loo. "Heuristics for scheduling in a flow shop with multiple processors." *European Journal of Operational Research* 113.1 (1999): 113-122.

12 Rajendran, Chandrasekharan, and Dipak Chaudhuri. "An efficient heuristic approach to the scheduling of jobs in a flowshop." *European Journal of Operational Research* 61.3 (1992): 318-325.

13 Moursli, Omar, and Yves Pochet. "A branch-and-bound algorithm for the hybrid flowshop." *International Journal of Production Economics* 64.1 (2000): 113-125.

14 Santos, D. L., J. L. Hunsucker, and D. E. Deal. "Global lower bounds for flow shops with multiple processors." *European Journal of Operational Research* 80.1 (1995): 112-120.

15 Fattahi, Parviz, et al. "A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations." *Applied Mathematical Modelling* 38.1 (2014): 119-134.

16 Gupta, Jatinder ND. "Two-stage, hybrid flowshop scheduling problem." *Journal of the Operational Research Society* (1988): 359-364.

17 Gupta, J. N. D., A. M. A. Hariri, and C. N. Potts. "Scheduling a two-stage hybrid flow shop with parallel machines at the first stage." *Annals of Operations Research* 69 (1997): 171-191.

18 Sriskandarajah, Chelliah, and Suresh P. Sethi. "Scheduling algorithms for flexible flowshops: worst and average case performance." *European Journal of Operational Research* 43.2 (1989): 143-160.

19 Brah, Shaukat A., and Luan Luan Loo. "Heuristics for scheduling in a flow shop with multiple processors." *European Journal of Operational Research* 113.1 (1999): 113-122.

20 Linn, Richard, and Wei Zhang. "Hybrid flow shop scheduling: a survey." *Computers & industrial engineering* 37.1 (1999): 57-61.

21 Kochhar, Sandeep, and Robert JT Morris. "Heuristic methods for flexible flow line scheduling." *Journal of Manufacturing Systems* 6.4 (1987): 299-314.

22 Ruiz, Rubén, and José Antonio Vázquez-Rodríguez. "The hybrid flow shop scheduling problem." *European Journal of Operational Research* 205.1 (2010): 1-18.

23 Voß, Stefan. "The Two—Stage Hybrid—Flowshop Scheduling Problem with Sequence—Dependent Setup Times." *Operations Research in Production Planning and Control*. Springer Berlin Heidelberg, 1993. 336-352.

24 Haouari, Mohamed, and Rym M'Hallah. "Heuristic algorithms for the two-stage hybrid flowshop problem." *Operations research letters* 21.1 (1997): 43-53.

25 Nowicki, Eugeniusz, and Czeslaw Smutnicki. "The flow shop with parallel machines: A tabu search approach." *European Journal of Operational Research* 106.2 (1998): 226-253.

26 Nowicki, Eugeniusz, and Czeslaw Smutnicki. "A fast taboo search algorithm for the job shop problem." *Management science* 42.6 (1996): 797-813.

27 Cheng, Jinliang, Yoshiyuki Karuno, and Hiroshi Kise. "A shifting bottleneck approach for a parallel-machine flowshop scheduling problem." *Journal of the operations research society of Japan* 44.2 (2001): 140-156.

28 Naderi, B., et al. "An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness." *Expert systems with Applications* 36.6 (2009): 9625-9633.

29 Jin, Zhihong, Zan Yang, and Takahiro Ito. "Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem." *International Journal of Production Economics* 100.2 (2006): 322-334.

30 Kurz, Mary E., and Ronald G. Askin. "Scheduling flexible flow lines with sequence-dependent setup times." *European Journal of Operational Research* 159.1 (2004): 66-82.

31 Zandieh, M., SMT Fatemi Ghomi, and SM Moattar Husseini. "An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times." *Applied Mathematics and Computation* 180.1 (2006): 111-127.

32 Mirsanei, H. S., et al. "A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times." *Journal of Intelligent Manufacturing* 22.6 (2011): 965-978.

33 Ruiz, Ruben, and Concepción Maroto. "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility." *European Journal of Operational Research* 169.3 (2006): 781-800.

34 Oĝuz, Ceyda, and M. Fikret Ercan. "A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks." *Journal of Scheduling* 8.4 (2005): 323-351.

35 Engin, Orhan, Gülşad Ceran, and Mustafa K. Yilmaz. "An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems." *Applied Soft Computing* 11.3 (2011): 3056-3065.

36 Guirchoun, Samuel, Patrick Martineau, and J-C. Billaut. "Total completion time minimization in a computer system with a server and two parallel processors." *Computers & Operations Research* 32.3 (2005): 599-611.

37 Shirazi, Arash N., Meghan Steinhaus, and Manbir Sodhi.”Modified Genetic Algorithm Based on Human Cell Mechanisms”. Manuscript in preparation.

38 Shirazi, Arash N., Meghan Steinhaus, and Manbir Sodhi.”Scheduling Flow Line by Fuzzy Cell Genetic Algorithm”. Manuscript in preparation.

39 Brooke, A. Kendrick, and A. D Meeraus. *GAMS release 2.25; a user's guide*. No. 519.76 B872. GAMS Development Corporation, Washington, DC (EUA), 1996.

TABLES

Table 1. Pseudocode of the Problem generator for flexible flow line data sets

Step	Description
1	**Initialization**
2	**Input** *n, s, m, L, U*
3	Makespan = 0
4	Generate sets with n number of jobs :$T = \{T_{1,1}, T_{1,2}, \ldots, T_{j,k}\}$
5	**For** each machine in parallel; $k = 1,2,\ldots,l$
6	Calculate the first order job $i \in T_{j,k}$; $Pt_{1,j} = random\ [L,U]$
7	$Makespan_k = Pt_{1,j}$
8	**For** each job in stage s ; $j \in T_{j,k} \mid i > 1$
9	Calculate the process time for job i in machine j $Pt_{i,j} = random\ [L,U]$
10	$Makespan_k = Makespan_k + Pt_{i,,j}$
11	$Pt_{i-1,,j+1} = Pt_{i,,j}$
12	**End For**
13	**End For**
14	**Set** $Makespan = \arg\max_{k \in 1,2,\ldots,l} Makespan_k$
15	**Save** $Process\ Time = [Pt_{i,j}]_{n \times m}$, $Makespan$
16	**End**

Table 2. Problem sets considered

	Machine 2	Machine 5	Machine 10
#Jobs	5,8,10,20,30,40,50,50,70,100	20,30,40,50,100	20,40,100
#Stage	2,5,15	20,30,40,50,100	10,20,40,60,100
Processing time	Random[1,10]	Random[1,10]	Random[1,10]
#Operation ($n \times s$)	10-300	40-10000	40-10000
#Instance per set	10	10	10
#Instance total	100	50	60

FIGURES

Fig. 1. Graphical representation of the problem design which *j*, *s* and *p* are index representative of jobs, stages and machines respectively.

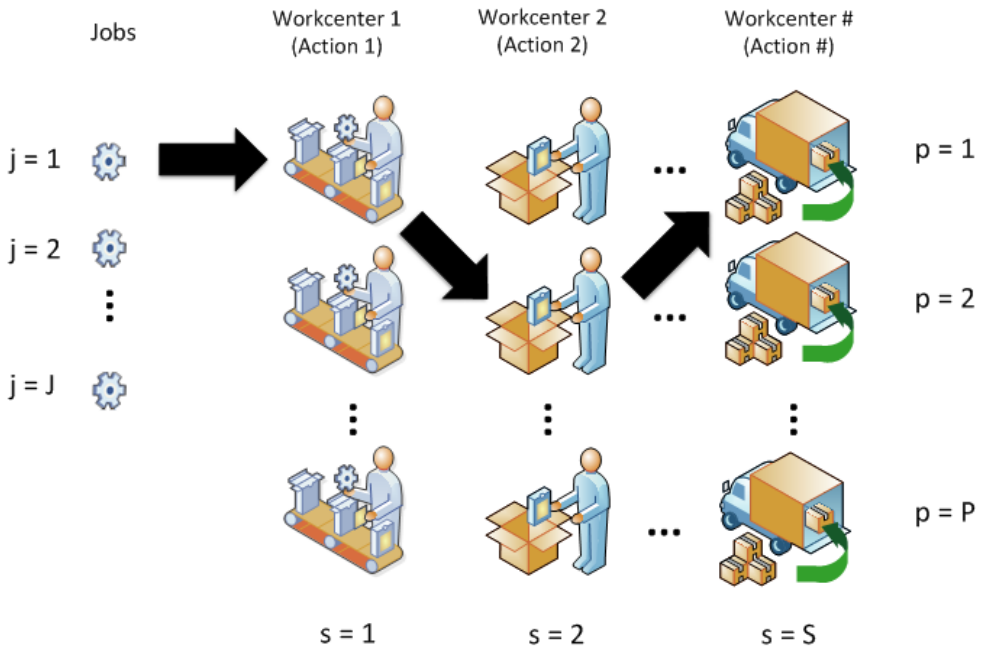


Fig. 2. FCGA flowchart: considers the proposed GA and fuzzy logic controller for adjusting the procedures in mitosis and meiosis.

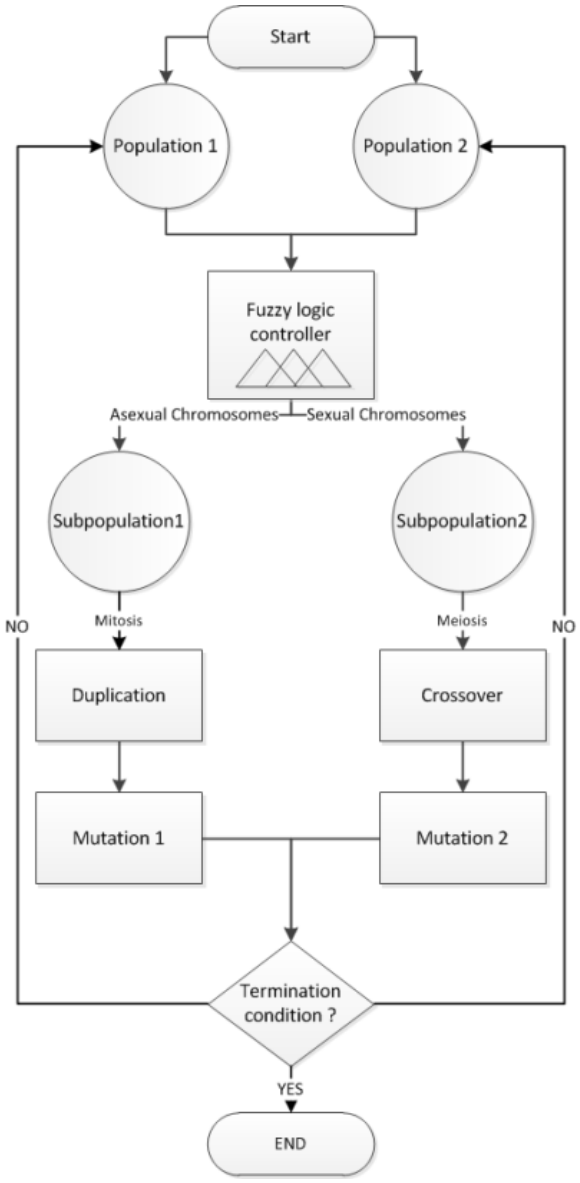


Fig. 3. An example of problem generator to provide a known solution (18sec) for 4 jobs 2 stages and 2 machines.

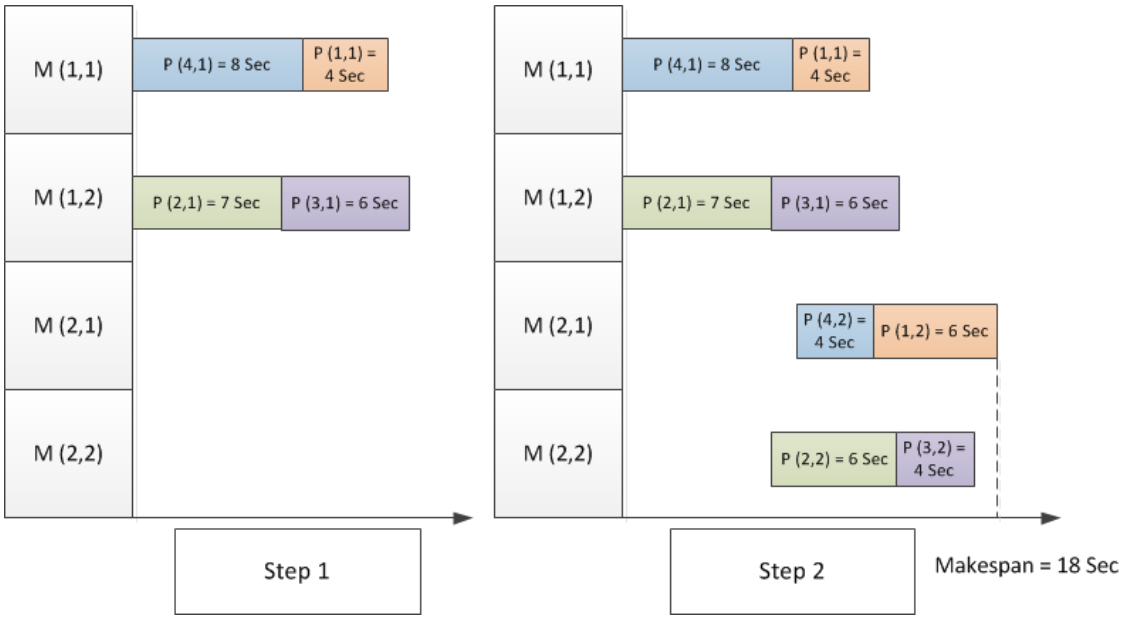


Fig. 4. Compare FCGA and problem generator solutions with IP results for measuring the algorithm performance on problems 5×2, 8×2, 10×5with 2 machines in each stage.

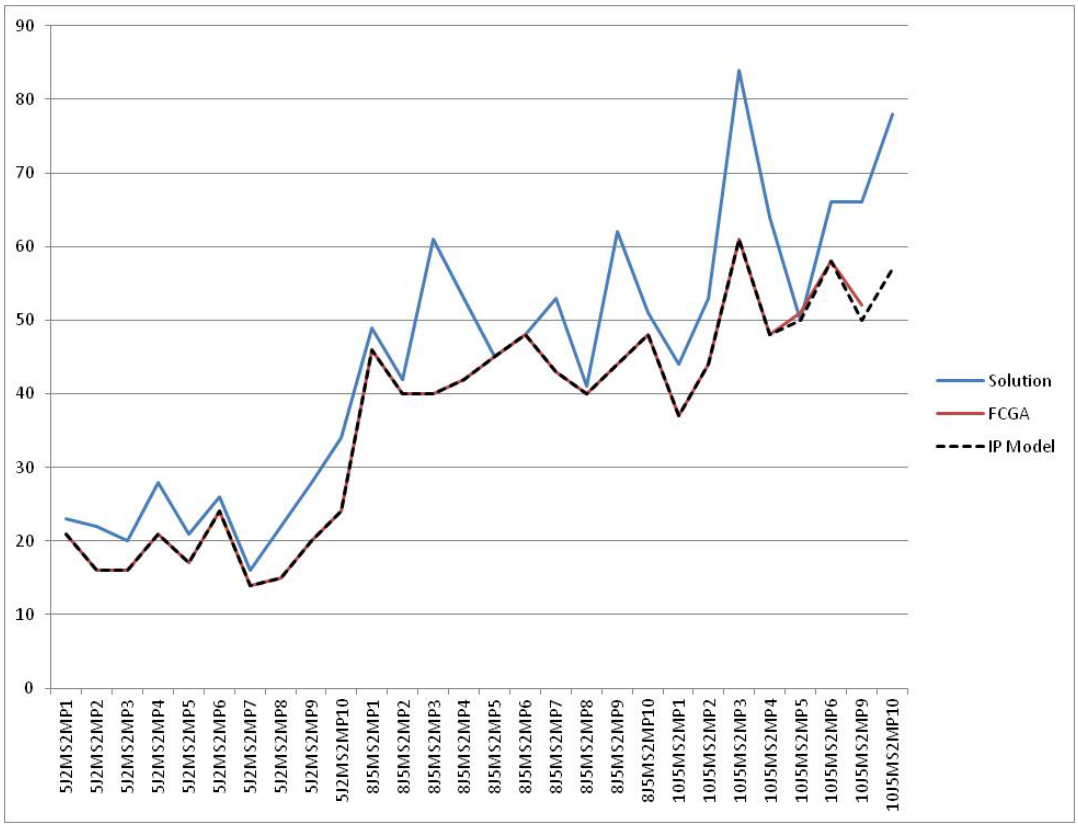


Fig. 5. Compare FCGA and problem generator solutions for measuring the algorithm performance on problems 20×5, 20×15, 30×5, 40×5, 50×5, 70×5, 100×5 with 2 machines in each stage.

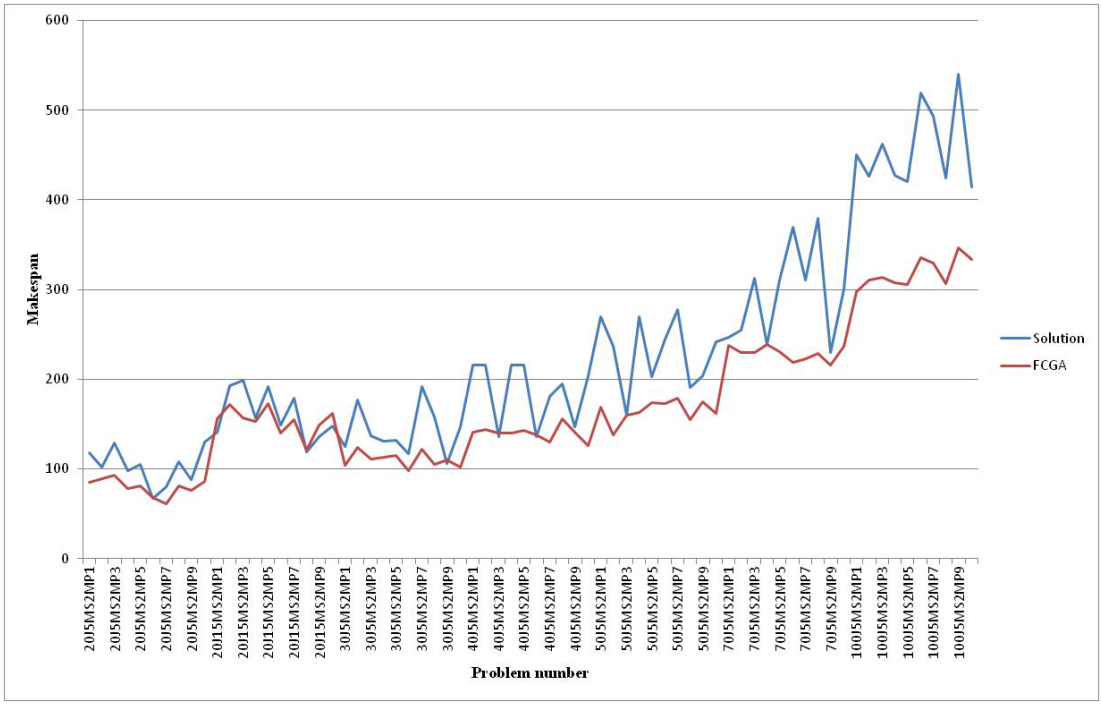


Fig. 6. Compare FCGA and problem generator solutions for measuring the algorithm performance on problems 20×20, 30×30, 40×40, 50×50, 100×100 with 5 machines in each stage.

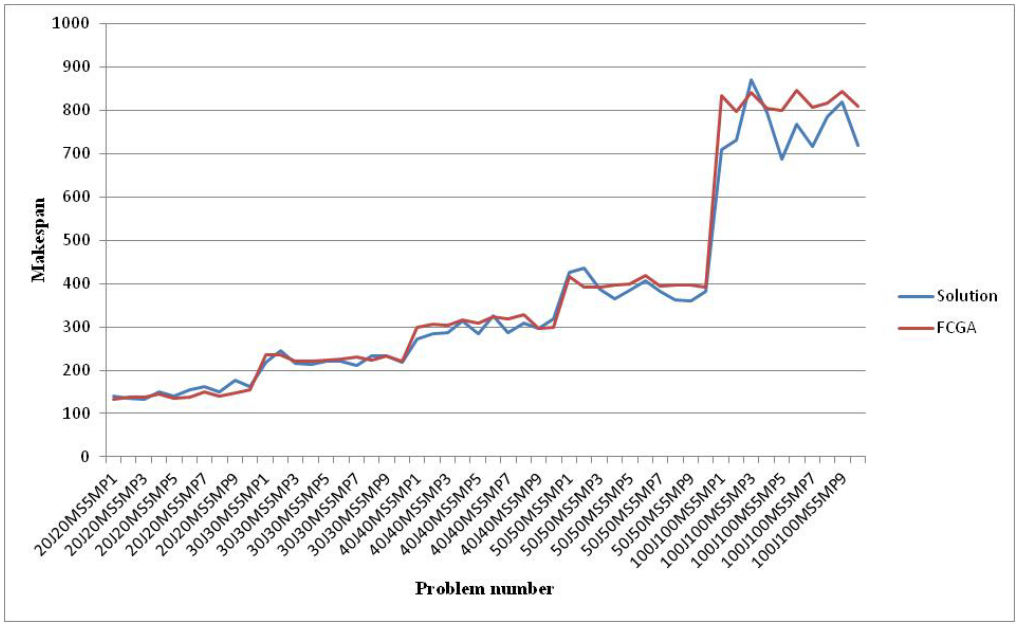
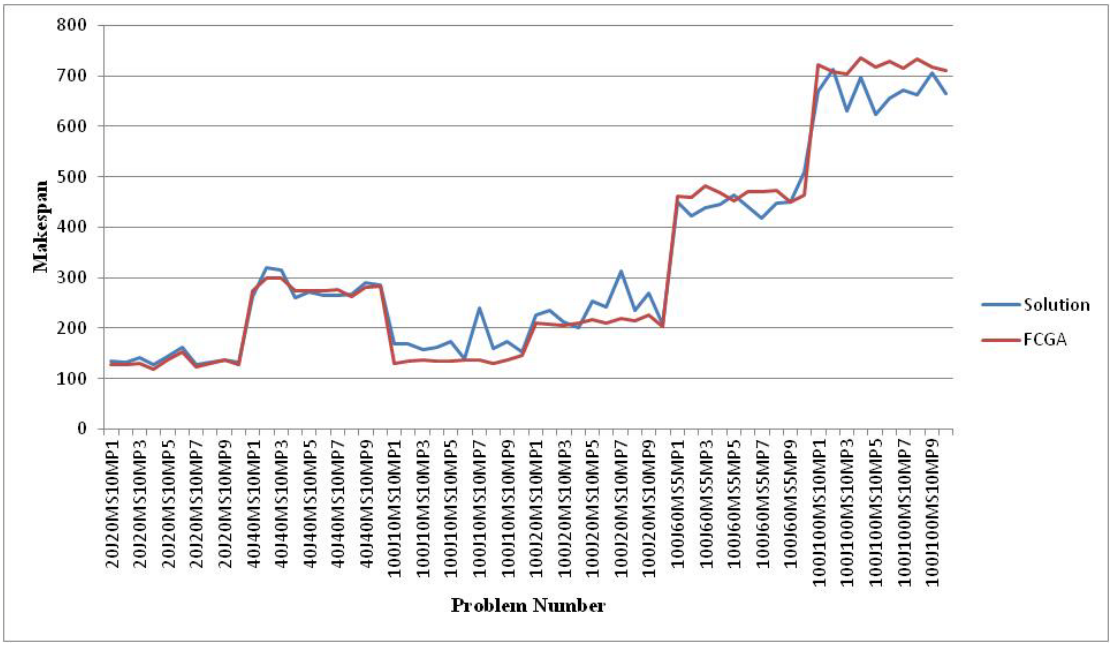


Fig. 7. Compare FCGA and problem generator solutions for measuring the algorithm performance on problems 20×20, 40×40, 100×10, 100×20, 100×60, 100×100 with 10 machines in each stage.



MANUSCRIPT – IV: MULTI-OBJECTIVE GENETIC ALGORITHM FOR BACKUP STATION IN FAULT-TOLERANT FLOW LINE DESIGN

In preparation for submission to International Journal of Production Research

Arash Nasrolahi Shirazi[1] and Manbir Sodhi[1]

[1]Department of Mechanical, Industrial and Systems Engineering, University of Rhode Island, Kingston RI 02881

Corresponding author: Arash Nasrolahi Shirazi

Department of Mechanical,

Industrial &Systems Engineering

University of Rhode Island

Kingston, RI 02881

E-Mail: arashshirazi@uri.edu

ABSTRACT

For automotive assembly, robots have been used to increase line productivity and efficiency and improve quality. However, robot failures reduce the throughput rate and product quality. There are several ways of recovering from line failures. One approach is to establish a manual backup station dedicated to processing those jobs that were incomplete when the line failed, allowing the line to re-start with fresh jobs at each station. Operations performed at this station usually take longer, and are not commensurate in quality with the automated stations. Another approach, developed in this paper, is to design a line with some redundancy, in-line backup stations and a manual recovery station. The backup stations are part of the main line but utilize versatile robots. In this case, a line failure is handled by reconfiguring the backup stations to perform as many make-up operations as possible, but the manual backup station is used for tasks that are not very demanding in complexity or precision. A multi-objective Genetic Algorithm approach for line design and for reallocating tasks when failure occurs is presented. This system is compared with an alternate approach that configures the line with a high level of redundancy and uses a backup station for all recovery. A comparable throughput is achieved with lower levels of redundancy and with fewer jobs sent for manual completion.

Keywords: assembly line system, robots failures, backup station method, robots redundancy, multi-objective genetic algorithm

INTRODUCTION

Assembly lines process parts on consecutive stations, with components or subassemblies merged, with precedence restrictions, to produce a complete product. A major advantage of these systems is that workers with limited training (specific to the station they are assigned to) can assemble complex products. Once set up, the line is very easy to manage – work flows linearly, and the supervisory tasks include maintaining the pace of the line, keep stations stocked with the necessary parts, and quality checks. High production rates can be attained using assembly lines- however, because assembly lines are customized to the product being produced, the initial investment required may be high. Because of the simplicity of operation and the high cost, assembly lines are operated with levels of utilization. However, the theoretical maximum utilization depends on the balanced allocation of work to the stations. The assembly line balancing problem (ALB) is complex, and many different scenarios have been considered by researchers [1-3].

Robotic assembly line problems

Although many generations of workers have toiled on assembly lines, robots are increasingly being considered for assembly tasks. Even though the dexterity of robots is vastly inferior to that of persons, highly automated lines are preferable when tasks require a high degree of precision. Robot assembly lines (rALB) have been extensively used the body and paint shops of the automotive industry [4]. Typically, four to five-thousand spot welds are performed by up to one hundred or even more welding robots in an advance automotive plant. This can require investments for

equipment exceeding one hundred million US dollars [5]. Although the use of robots in these systems offers several advantages, including increased productivity, highly automated and complex systems may face failures and machine breakdowns that can lead to the shutdown the entire line [6]. Besides decreased productivity during the delay, in-process products can also be damaged or require rework due to robot failures. Although the Assembly Line Balancing approach can be used for configuring rAls as well, there are idiosyncrasies in this application that require reformulation of the approach first proposed, and well developed since, by Helgeson et al [7].

Many different optimization approaches have been proposed for solving ALBs. These include dynamic programming, integer programming and branch-and-bound methods [8-10]. However, when dealing with applications requiring the configuration of lines with up to several thousand tasks, and constraints including positioning choices for task allocation, the computational resources required for optimal solutions are not practical. ALB problems are NP-Hard [11]. To solve problems with reasonable computing time, efficient heuristic or metaheuristic approaches are preferable. There have been significant attentions for solving ALBs problems with Genetic Algorithms (GAs) among various metaheuristic approaches. The reasons for effectively applying GAs for ALBs problems are stochastic search and optimization techniques base on ideology from evolutionary theory [12]. The first application of GAs for ALBs problem was proposed by Falknauer and Delchambre [13]. Afterwards many researchers have concentrated on implementing GAs for ALBs in different ways but simple version of the problem which considered single objective and ignored the complexity of the real life models [14-16]. The fundamental element for solving more

practical ALBs models such as mix-model production, u-shape lines and rALBs is to formulate them in a multi-objective ways. Using multi-objective Genetic algorithm for solving rALB problems assist researchers to optimize cycle time, number of stations and the workload in each station simultaneously [17-20]. Technically, in typical layouts of the body welding shop each line contains different robotic stations. Each station has several welding robots. Robots are equipped with various welding guns working simultaneously [21]. Some robots have more than one degree of freedom allowing them to cover more welding spots. This flexibility of robots let them to do more than one tasks in the same cell or station.

There are specific terms for assembly line balancing problems. For instance, operation is a part of the work in an assembly process. A station is known as a division of assembly line where a number of different operations are completed. The products precedence constraint is a type of technical or organizational restrictions between the operations to observe the priority of the tasks for the production process. The precedence of the tasks is being shown as a graph connecting the tasks nodes to each other. The cycle time is defined as a maximum amount of time that a work piece can be processed by a station to meet the customer demand. Ideal time is a difference between cycle time and the station time [22]. Most of the time assembly tasks can be performed on both sides of the line. While some tasks are optimized to be carried out at one of the two sides, others can be performed on either side of the line. Therefore, tasks can be divided into different groups: L (left), R (right), and E (either)-type tasks. Two-side assembly line was proposed previously [23].

Model description

The basic of presented model is generalized assembly line balancing problems for automated assembly lines. The key objective is minimizing the cycle time of the system. Second objective is minimizing the idle time in each station. Moreover, the number of stations needs to be minimized to reduce the redundancy of robots in each station. The set of welding spot groups is divided in to subsets. Each group of welding spots is associated with a label ($S_{p,d}$) where p is the position of the welding spot in each station (rear (R), Front (F), or either (E)), and d is the direction of operations (left (L), right (R) or either (E)) (Fig. 1).

Genetic Algorithm approach

In order to employ GAs for proposed model should start with a set of primitive feasible solutions (population). There are six important steps to follow in GAs:

Step 1: Randomly generate chromosomes which represent the solutions for the problem. Encoding the solutions for the problem is to manipulate the chromosomes into feasible problem solutions. Each chromosome has a specific weight which defined by fitness function.

Step 2: Randomly select two solutions (parents) from the population. Use crossover to generate new solution (offspring) which inheritance of properties from both parents.

Step 3: Mutation is the way to generate diversity in a pool of solutions. Specifically, mutation is chooses one piece of information in the chromosome

randomly and replaces with different information. This is necessary to check feasibility of chromosome and structured as a feasible solution for our problem after the mutation.

Step 4: Use fitness function to calculate the value for each offspring.

Step 5: Finally, the quality of offspring with the worse fitness value in the population will be checked. If the offspring offers a higher quality, the offspring will be kept as new parents in population and the worse quality parents in population will be removed.

Step 6: Termination is the time when n replications of steps 2-5 cannot improve the solutions. To implement GAs with six procedures above in rALB problem should follow 3 sections.

Primitive GAs procedures

This part starts with the main vector containing a sequence of tasks. The orders of these tasks are based on assembly line precedence sequence. The other subvectors are assigning tasks to the stations and guns that will be used in each station by robots. An example of precedence diagram for problem and the vector of tasks sequence which is used in genetic process such as crossover, mutation and selection have showed in Figure 2. This needs to be noted that, the solutions generated randomly. Thus, there is a high possibility for the main vectors to be non-feasible solutions. The main vectors which consist of the sequence of activities (welding spots) have to be

feasible for the problem (based on the precedence diagram) before starting other steps in GAs.

Here, we try to show the procedure of transforming non-feasible solution to the feasible one by using decision tree. The main vector that consists of random sequence of tasks should transfer the tasks to the decision tree from top to bottom in order of left to the right. This would be helpful to check the root tasks for having higher priority than their leaves tasks. If the task in the root has a lower priority than the task in the leaf, the task will be swapped with each other. In Figure 3, task 2 has higher priority than task 4 and task 5. So, first task 2 is replaced by task 4 and the second swap place task 2 in a higher level of priority than tasks 4 and 5.

At the end of this process the main vector has a feasible sequence of tasks. Population consists of different vectors with feasible solutions.

Crossover and mutation

In crossover, two parents have been selected from the population to create a new offspring. The offspring inherits information from both parents. It is necessary to check the feasibility of the offspring as well.

The proposed crossover for this problem works as follows:

a) Choose two parents from population and cut them in 3 parts. One body and two partitions (left and right) as leaves.

b) Offspring tree is defined as a body part from parent 2 and the subset of leaves from parent 1.This should be noted that the left and right partitions have the same roots.

c) Check the feasibility of the offspring tree and fix the tree.

In the following example the offspring's elements which inherit from parents 1 and 2 are marked with bold font.

Parent 1: 1 2 3 5 6 **8 9 4 7** 10
Parent 2: 1 2 3 5 8 **4 7 6 9** 10
Offspring: 1 2 3 5 8 8 9 4 7 10
Feasible offspring: 1 2 3 5 8 6 9 4 7 10

To check the feasibility of offspring, the repeated and missed task in a tree needed to be found and get replaced by the redundant task in a feasible place. The procedure of crossover and how to turn the non-feasible offspring to the feasible one has been shown in Figure 3.

The procedure in mutation is to select the position of one task randomly and put a different task in random. To make the offspring feasible, similar steps to crossover feasibility procedure need to be followed (Fig. 4).

Decoding procedure

After crossover and mutation, decoding procedures is the final step to evaluate that whether the results are improved or not. These procedures are illustrated in the following steps:

1. Separate the sets of spots in different stations.

2. Assigned the sets of spots in each station to feasible robots according to their locations and capable guns in each station.
3. Evaluate the cycle time for the given balance.
4. Calculate the workload (idle time) in each station.
5. Count the number of sets which has been done by manual backup.
6. Assign weight for offspring and if this is better than the results in population replace it with the worst result.

MULTI-OBJECTIVE ADAPTIVE- WEIGHT GENETIC ALGORITHM

Non-dominance is the key concepts in multi objective problems. For minimization problem which has more than one objective functions (f_K *,for* k *=1,2,...,m|* $m \geq 2$) if x_1 and x_2 are two possible solutions then x_0 is dominate to x_1 and if only both results are satisfying the following conditions:

$$f_k(x_0) \leq f_k(x_1) \ for \ k = 1,2,..,n$$
$$f_l(x_0) \leq f_l(x_1) \ for \ l = 1,2,..,n$$

In equation 1, the result of x_0 is a privileged solution compared to x_1 for all objectives. Equation 2 shows the solution x_0 is specifically less than x_1 in at least one objective, or If any of these two conditions has violated, then we could say x_0 is not dominate to the solution x_1 . In other words, we can say x_0 is not dominated by any objectives individually then, x_0 is non-dominated solution [24]. To solve Multi Objective problem by using genetic algorithms, adaptive–weighted Genetic Algorithms (awGA) is applied [22]. aWGA is using the information from current population and calculate the weights in order to search toward possible non-dominated

solutions. In aWGAs, the fitness function is $z = \sum_{i=1}^{n} p_i\left(\left|f_i(x) - f_i^{min}\right|\right)$ where $f_i(x)$ is i^{th} objective and p_i is a i^{th} weight that has assigned to the function and N is the total number of objectives in the problem. To calculate p_i we used $p_i = \frac{1}{f_i^{max} - f_i^{min}}$ for $i = 1, 2, 3, \ldots, N$ number of objectives. At the first part of the problem objectives are the cycle time, the total work load stations and the number of stations. Ten replications were used for each case to find the maximum and minimum objectives. Finally, a proposed fitness function was introduced (z) to find the optimum solution for the first part of the problem. The pseudocode of aWGAs for our model is exhibited in Table 1.

DIFFERENT APPRAOCHES DURING FAILURES

This body-shop consists of 8 stations and 31 robots with different capability for 40 different guns. It should be noted that the capability of the robot to perform an operation are mainly determined by its gun configuration and physical location of each robot in a station.

Obviously, the robots with high capability of doing different tasks in using guns are more prone to failure than the other robots. Therefore, different case studies have proposed different solutions for robot failures [25-26]. One of the primitive solutions is to use manual backup (MB) station at the end of body-shop. The MB station is utilized to perform the job in a case that none of the robots are capable to cover the failed robot's position. Assumption for this part is that MR station is always feasible and in order to avoid stoppage this station must be used. But the group of spot which has done with MB station has lower quality and they take longer time to finish their jobs than robotic solution (task which done by robot). Consequently, using MB

station will increase the cycle time. It should be noted that the current use of the term ‘backup station’ has a different meaning from previous studies [26].

They also proposed the solution to maximize the system level of redundancy in order to make the line more robust against robots’ failures (Figure 6). This has been shown that the higher level of redundancy for robots provided less chance to use MB station and shorter cycle time in assembly line by covering the failed robot.

However, this system suffers from two main problems which are not pleasant in a real scenario. First, there is no doubt that increasing the level of redundancy in assembly line prevents stoppage in the body-shop. However, using a high level of redundancy means that a lot of money was invested to equip our robots to afford less than 50% of the performance of our robots. Second, in previous studies they have neglected the precedence relationships for tasks. Typically, precedence relationships can be resulted from the product structure along with the characteristics of the production system. For instance, as it is shown in Figure 6, the robot 2 in station 2 performs the group of tasks from robot 2 in station 1 in addition to its own group of tasks. If the group of tasks in station 1 need to be done before entering to the station 2, then, this will not be feasible to assign the group of task to the station 2.

In this paper, we have tried to make our model near to a real life condition. Using a backup station for just the robots which are prone to failure could be considered as an advantage. Moreover, the majority of stations are located next to the main station. They can decrease the high level of redundancy and follow the precedence relationship of tasks.

In this study, the backup station model is proposed to prevent excessive unnecessary redundancy in the system and follow the precedence relationship of the tasks in body-shop during the fail time. Here, those robots that are carrying many guns to perform different group of spots have a higher robot capacity (RC) in comparison with the other robots at a same station. So, they are more prone to failure than the other robots. To backup these robots in a case of failure, they have selected a backup station. For instance, as it is exhibited in Figure 7 the robot R1 is performing 12 groups of tasks and robot R4 carrying out 7 groups of tasks but the robot R2 and the R3 are both doing only 2 groups of tasks.

Evidently, robots R1 and R4 are more susceptible to failure than robots R2 and R3. Since it would be difficult to back up their tasks in a case of failure, thus, they should locate backup robots in a station neighboring to the main station.

Formulation for choosing robots in backup station is:

$$BS_m = \left\{ \sum_{r \in R} R_{m,r} \,\middle|\, R_{m,r} \geq RC_{max} \right\} \qquad \forall\, m \in M$$

To find maximum robot capability ($\mathrm{RC_{max}}$) in each station we have:

$$RC_{max} = \max\left(\sum_{s \in S} \sum_{j \in J} x_{m,r,s} \times y_{m,r,j} \right) \times 50\% \quad \forall\, r \in R, m \in M$$

We should consider that the average level of redundancy for BS model is 2.5 which is far better than the level of redundancy of 4 and 6.

As it was noted before, here, the three considered performance measures are the cycle time, the total idle time in stations and the quality. The percentage of groups with failed robots that are back up by MB station is the quality measurement for each

scenario. Kahan et al.(2009) findings showed that the reason of low quality in MB station is the quality of welding spots which performing by robots are much higher than the MB station quality. Moreover, using MB station has a negative effect on the cycle time which increases the cycle time by performing the spots three times higher than the robotic time.

RESULTS AND DISCISSION

First step is to minimize the cycle time with minimum workload in minimum stations. For the first part, 8 stations and 31 robots is an optimized package to perform 72 different groups of spots with 40 different guns. The capability of robots was elevated by increasing their guns up to 4 and 6 average level of redundancy. However, for BS scenario the high capacity robots are located in backup station next to the main stations.

The assembly line composed of 4 level of redundancy (4LR), 6 level of redundancy (6LR) and Backup station (BS) in total three different solution approaches for reallocation problems.

The study showed that the average of cycle time, the total workloads in all stations and the number of group of spots were carried out by the MB station. To evaluate the quality line, two elements were considered including the number of groups reordered to the MB station and the percentage of reordered groups. The assumption for BS stations is based on the fact that the robots in BS station wouldn't fail to perform their tasks but their work performance may change during failures. As

we mentioned before, BS redundancy is 2.5 which is extremely low compared to the level of redundancies 4 and 6 in a system.

The average cycle time was collected for two different scenarios. First scenario was simulated failure for the most susceptible robots (robots with high RC) fail to operate in our system. In this case, BS station shows the best average cycle time of 92.8 second during 4 different critical situations in the system. The BS average cycle time is 10.3% less than the system with the level of redundancy of 6 (6LR) and definitely superior result in compare with the system with redundancy of 4 (4LR) by 18.9 %.

The collected results in the same critical failure situations showed an average idle time of 96.2 second in the system which implemented by BS versus 119.8 second of the 6LR. The 4LR shows poor results of average idle time by 202.2 second (Fig.8).

Second scenario was based on the 16 random failures in a system. The 6LR is the second best in terms of the average cycle time performance of 104.3 second. However, BS with the average cycle time of 103.7 second was the optimal case (Fig.9). Furthermore, the average cycle time for 6LR increased by 30.8 % in compared to BS one.

In this case, we compare the ratio percentage of the reallocated groups of spots by MS station to the total reallocated groups of spots.

In this case the 6LR and BS show relatively similar results. The BS shows that the reduction of low quality just by 2.5%.However, the high level of redundancy

between the system with 6 level of redundancy and BS with just 2.5 level of redundancy in a system is considerable. The 4LR still shows a low quality level of production in the system by 24.1%.

The simulation results have compared between BS station performance during most critical failures and other situation with less cycle time in all situations. The quality of products in BS system for other failures shows much better results than the system with 4 level of redundancy and approximately similar to the one with the redundancy level of 6.

CONCLUSION

In this work we purposed a new solution of back up station for spot welding reallocation problem due to robot's failures. The BS solution can not only prevent the stoppage during the failure with less cycle time and idle time in the assembly line system with less level of redundancy, but also follow the production procedure of precedence tasks in a system perfectly.

ACKNOWLEDGMENT

This work was supported by University Of Rhode Island partially.

REFERENCES

1. A. Scholl, N. Boysen, and M. Fliedner, “The sequence-dependent assembly line balancing problem” OR Spectrum, vol. 30,pp. 579-609, March 2008.
2. A. SHTTJB, “The effect of incompletion cost on line balancing with multiple manning of work stations.” THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH vol. 2 , pp. 235-245,1984.
3. A. Scholl. Balancing and Sequencing of Assembly Lines. Heidelberg; New York: Physica-Verlag, 1999.
4. A.E. Owen, Assembly with Robots. Kogan Page Ltd, 1985.
5. Spieckermann, S., K. Gutenschwager, H. Heinzel, and S. Voß. 2000. “Simulation based optimization in the automotive industry - A case study on body shop design.”Simulation 75 (5-6): 276–286.
6. Boysen, N., M. Fliedner, and A. Scholl. 2008. “Assembly line balancing: Which model to use when?.” International Journal of Production Economics 111 (2): 509–528.
7. Helgeson, W. B., M. E. Salveson, and W. W. Smith. "How to balance an assembly line." Management Report 7 (1954).
8. Salveson, Melvin E. "The assembly line balancing problem." Journal of Industrial Engineering 6, vol. 3 , p.p. 18-25,1955.
9. Bowman, Edward H. "Assembly-line balancing by linear programming." Operations Research 8, no. 3 (1960): 385-389.

10. Held, Michael, Richard M. Karp, and Richard Shareshian. "Assembly-line balancing—dynamic programming with precedence constraints." Operations Research 11, no. 3 (1963): 442-459.
11. Karp, Richard M. Reducibility among combinatorial problems. Springer US, 1972.
12. Gen, M.,& Cheng, R.(200). Genetic Algorithm and Engineering Optimization, New York:John Wiley & Sons.
13. Rubinovitz, J., and Gregory Levitin. "Genetic algorithm for assembly line balancing." International Journal of Production Economics 41.1 (1995): 343-354.
14. Kim, Yeo Keun, Yong Ju Kim, and Yeongho Kim. "Genetic algorithms for assembly line balancing with various objectives." Computers & Industrial Engineering 30.3 (1996): 397-409.
15. Leu, Y. Y., Matheson, L. A., & Rees, L. P. (1994). Assembly Line Balancing Using Genetic Algorithms with Heuristic-Generated Initial Populations and Multiple Evaluation Criteria*. Decision Sciences, 25(4), 581-605.
16. Sabuncuoglu, I., Erel, E., & Tanyer, M. (2000). Assembly line balancing using genetic algorithms. Journal of intelligent manufacturing, 11(3), 295-310.
17. Haq, A. N., Rengarajan, K., & Jayaprakash, J. (2006). A hybrid genetic algorithm approach to mixed-model assembly line balancing. The International Journal of Advanced Manufacturing Technology, 28(3-4), 337-341.
18. Levitin, Gregory, Jacob Rubinovitz, and Boris Shnits. "A genetic algorithm for robotic assembly line balancing." European Journal of Operational Research 168.3 (2006): 811-825.

19. Tasan, Seren Ozmehmet, and Semra Tunali. "A review of the current applications of genetic algorithms in assembly line balancing." Journal of intelligent manufacturing 19.1 (2008): 49-69.
20. Rekiek, Brahim, et al. "State of art of optimization methods for assembly line design." Annual Reviews in Control 26.2 (2002): 163-174.
21. Bartholdi, J. J. "Balancing two-sided assembly lines: a case study." THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 31, no. 10 (1993): 2447-2461.
22. Gen, Mitsuo, Runwei Cheng, and Lin Lin. Network models and optimization: Multiobjective genetic algorithm approach. Springer, 2008.
23. Bartholdi, J. J. "Balancing two-sided assembly lines: a case study." THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 31, no. 10 (1993): 2447-2461.
24. Augugliaro, Antonino, Luigi Dusonchet, and Eleonora Riva Sanseverino. "Evolving non-dominated solutions in multiobjective service restoration for automated distribution networks." Electric Power Systems Research 59.3 (2001): 185-195.
25. Kahan, Tomer, Yossi Bukchin, Roland Menassa, and Irad Ben-Gal. "Backup strategy for robots' failures in an automotive assembly system." International Journal of Production Economics 120, no. 2 (2009): 315-326.
26. Shin, Frank, Bala Ram, Aman Gupta, Xuefeng Yu, and Roland Menassa. "A decision tool for assembly line breakdown action." In Proceedings of the 36th conference on Winter simulation, pp. 1122-1127. Winter Simulation Conference, 2004.

27. Müller, Christoph, Thomas S. Spengler, and Manbir S. Sodhi. "Robust Flowline Design for Automotive Body Shops." IIE Annual Conference. Proceedings. Institute of Industrial Engineers-Publisher, 2014.

28. Gupta, Ashish, Gopalakrishna K. Shastry, Narahari K. Hunsur, Wayne Cai, and Robert Tilove. "An Approach for Automated and Optimized Selection of Weldguns for Spot Welding in Automotive Body Shop." Transactions of the ASME-B-Journ Manufacturing Science Engineering 134, no. 3 (2012): 034501.

TABLES

Table 1. aWGA pseudocode for the proposed problem

---

procedure: aWGAs procedure

input: the objectives cycle time $C(v_i)$, total workloads in stations $W(v_i)$

and number of stations $S(v_i)$, $\forall\, i \in N(Population\ size)$

output: evaluate fitness value for chromosome (v_i), $\forall\, i \in N(Population\ size)$

begin

$\{C^{max}\} \leftarrow \mathbf{max}\ \{C(v_i)\}$ $\forall\, i \in N(Population\ size)$

$\{W^{max}\} \leftarrow \mathbf{max}\ \{\mathrm{W}(v_i)\}$ $\forall\, i \in N(Population\ size)$

$\{S^{max}\} \leftarrow \mathbf{max}\ \{\mathrm{S}(v_i)\}$ $\forall\, i \in N(Population\ size)$

$P_1 \leftarrow \frac{1}{C^{max} - C^{min}}$

$P_2 \leftarrow \frac{1}{W^{max} - W^{min}}$

$P_3 \leftarrow \frac{1}{S^{max} - S^{min}}$

$eval\ (v_i) \leftarrow \{P_1(C(v_i) - C^{max}) + P_1(W(v_i) - W^{max}) + P_1(S(v_i) - S^{max})\}\ \forall\, i \in N$

output eval(v_i), $\forall\, i \in N$

end

FIGURES

Fig. 1. The position of welding guns and direction their performance

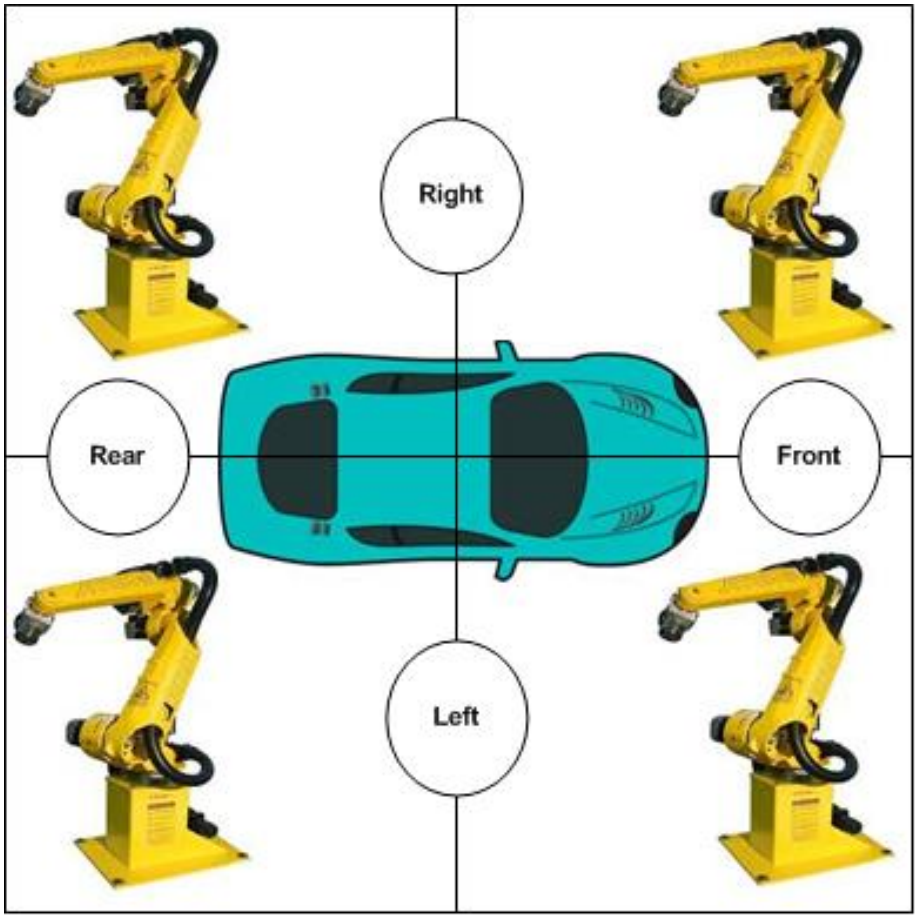


Fig. 2. An example of precedence diagram and main vector of tasks sequence

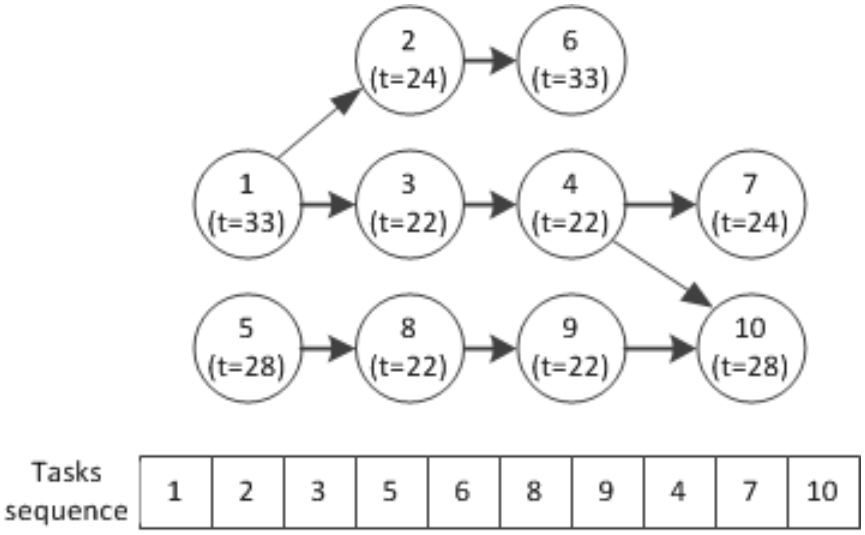


Fig. 3. Crossover procedure to generate feasible offspring

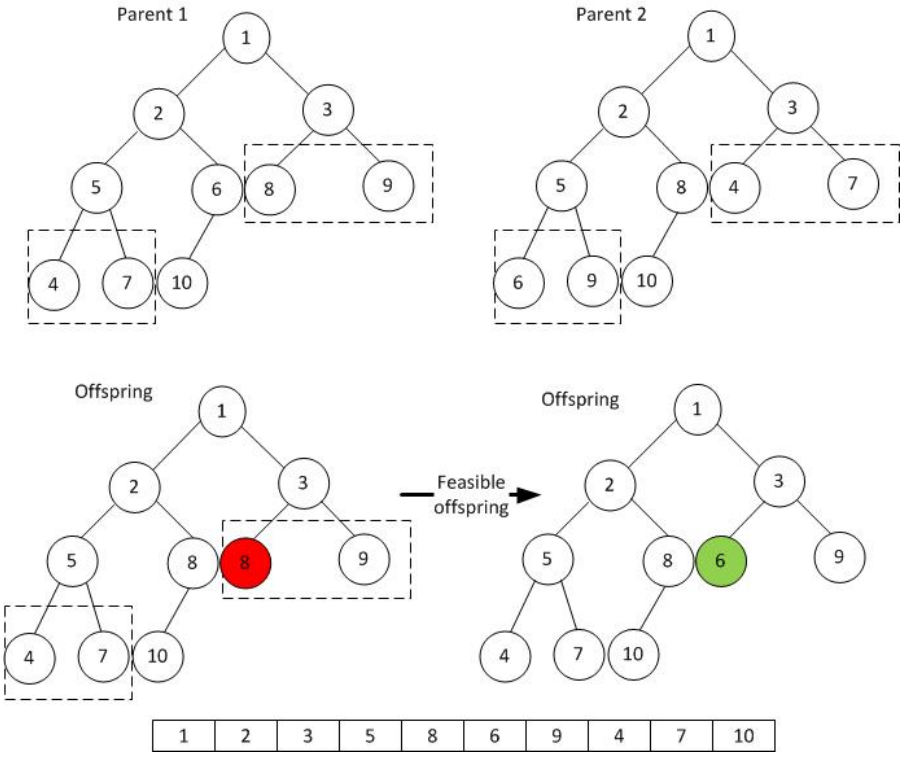


Fig. 4. Procedure in mutation by swapping the tasks (the top vector is infeasible and the bottom one is feasible task sequences)

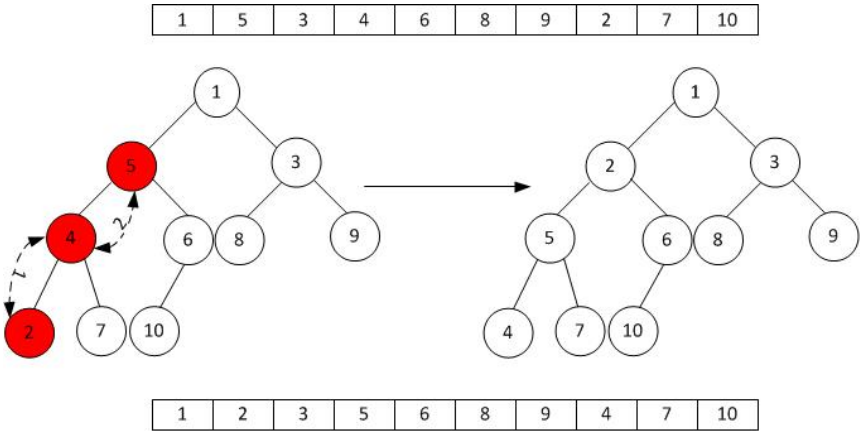


Fig. 5. Backup robot 2 in station 2 for failed robot 2 in station1

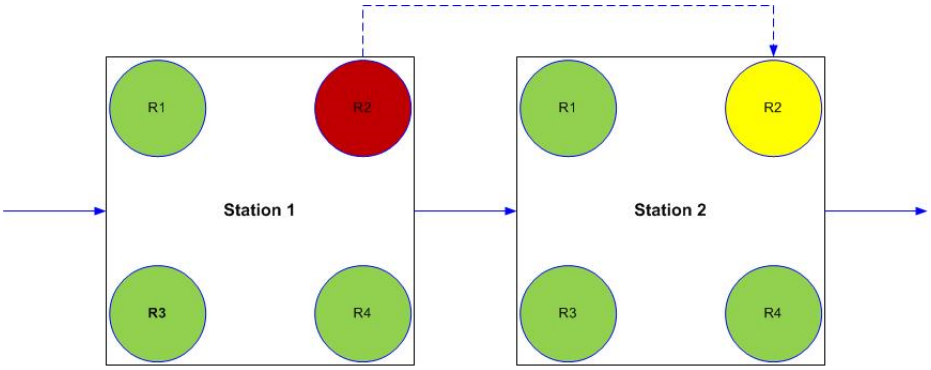


Fig. 6. Schematic of Back up station for high capability robots

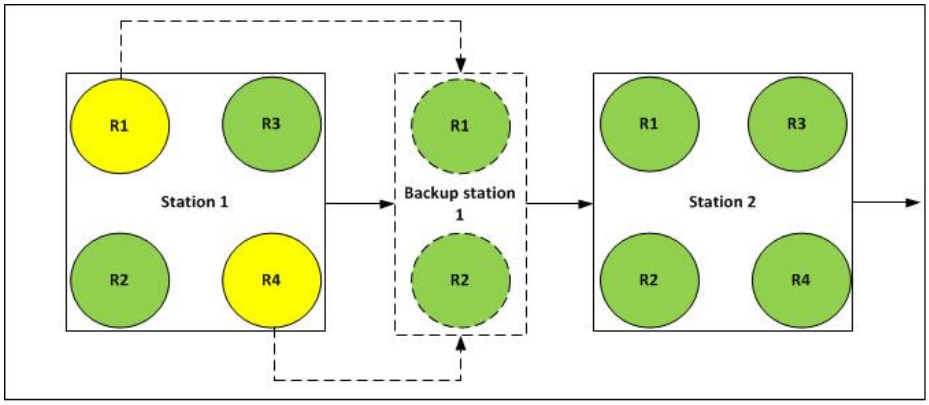


Fig. 7. Cycle time versus idle time for critical failures

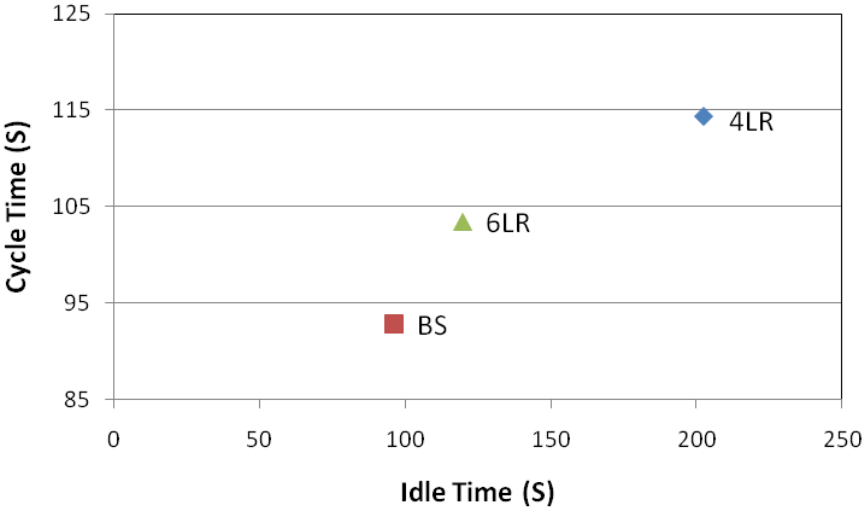
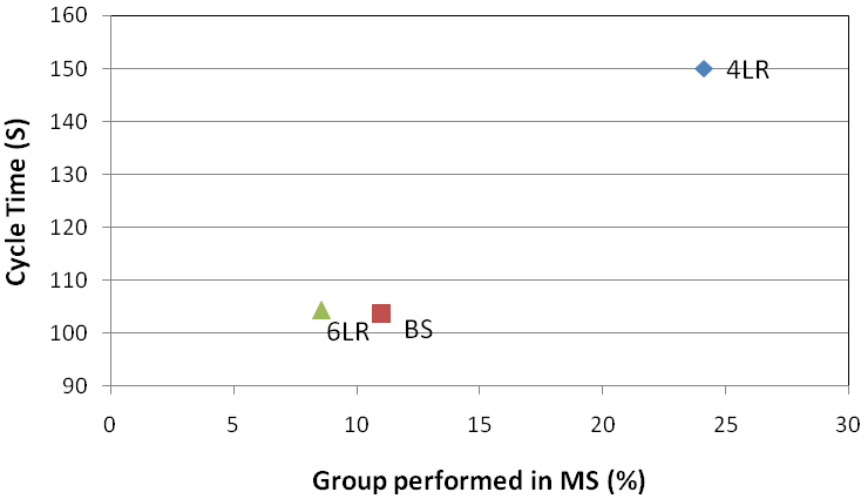


Fig. 8. Cycle time versus quality percentage



BIBLIOGRAPHY

1. H. John “Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial inteliigence” Oxford, England: U Michigan Press. vol. viii, pp. 183, 1975.
2. D. E. Goldberg “Genetic algorithms in search, optimization, and machine learning”.Reading, MA: Addison-Wesley, 1989.
3. E Alba and M Tomassini "Parallelism and Evolutionary Algorithms", IEEE Transations on Evolutionary Computation, vol. 6, no. 5, 2002.
4. T. C. Belding, “The distributed genetic algorithm revisited,” in Proc. 6th Int. Conf. Genetic Algorithms, L. J. Eshelman, Ed., 1995, pp. 114–121.
5. ___,“Distributed genetic algorithms,” in ICGA-3, J. D. Schaffer, Ed., 1989, pp. 434–439.
6. S. Baluja, “Structure and performance of fine-grain parallelism in genetic search,” in Proc. 5th Int. Conf. Genetic Algorithms, S. Forrest, Ed.,1993, pp. 155–162.
7. ___,“Improving flexibility and efficiency by adding parallelism to genetic algorithms,” Statist. Comput., vol. 12, no. 2, pp. 91–114, 2002.
8. V. S. Gordon and D. Whitley, “Serial and parallel genetic algorithms as function optimizers,” in Proc. 5th Int. Conf. Genetic Algorithms, S. Forrest, Ed., 1993, pp. 177–183.
9. Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer Science & Business Media, 1996.
10. Pornsing, Choosak, Manbir S. Sodhi, and Bernard F. Lamond. "Novel self-adaptive particle swarm optimization methods." Soft Computing (2015): 1-15.

11. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE Trans. Evol. Comput., vol. 6, pp.182 -197, 2002 .
12. Srinivas, M., and Lalit M. Patnaik. "Adaptive probabilities of crossover and mutation in genetic algorithms." Systems, Man and Cybernetics, IEEE Transactions on 24, vol. 4, pp. 656-667, 1994.
13. Andrea Rossi , Alessio Puppato & Michele Lanzetta (2013) Heuristics for scheduling a two-stage hybrid flow shop with parallel batching machines: application at a hospital sterilisation plant, International Journal of Production Research, 51:8, 2363-2376
14. Nawaz, Muhammad, E. Emory Enscore, and Inyong Ham. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem." *Omega* 11.1 (1983): 91-95.
15. Potts, Chris N., David B. Shmoys, and David P. Williamson. "Permutation vs. non-permutation flow shop schedules." *Operations Research Letters* 10.5 (1991): 281-284.
16. Herroelen, Willy, Bert De Reyck, and Erik Demeulemeester. "Resource-constrained project scheduling: a survey of recent developments." *Computers & Operations Research* 25.4 (1998): 279-302.
17. French, S. "Sequencing and scheduling, mathematics and its applications." (1982).
18. Rossi, Andrea, and Michele Lanzetta. "Scheduling flow lines with buffers by ant colony digraph." *Expert Systems with Applications* 40.9 (2013): 3328-3340.

19. Fisher, Henry, and Gerald L. Thompson. "Probabilistic learning combinations of local job-shop scheduling rules." *Industrial scheduling* 3 (1963): 225-251.
20. Brah, Shaukat A., and John L. Hunsucker. "Branch and bound algorithm for the flow shop with multiple processors." *European Journal of Operational Research* 51.1 (1991): 88-99.
21. Johnson, Selmer Martin. "Optimal two-and three-stage production schedules with setup times included." *Naval research logistics quarterly* 1.1 (1954): 61-68.
22. Taillard, Eric. "Some efficient heuristic methods for the flow shop sequencing problem." *European journal of Operational research* 47.1 (1990): 65-74.
23. Ruiz, Rubén, and Concepción Maroto. "A comprehensive review and evaluation of permutation flowshop heuristics." *European Journal of Operational Research* 165.2 (2005): 479-494.
24. Pan, Quan-Ke, and Rubén Ruiz. "A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime." *Computers & Operations Research* 40.1 (2013): 117-128.
25. Chakraborty, Uday Kumar, and Dipak Laha. "An improved heuristic for permutation flowshop scheduling." *International Journal of Information and Communication Technology* 1.1 (2007): 89-97.
26. Nowicki, Eugeniusz, and Czeslaw Smutnicki. "A fast taboo search algorithm for the job shop problem." *Management science* 42.6 (1996): 797-813.
27. Lin, S-W., and K-C. Ying. "Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems." *International Journal of Production Research* 47.5 (2009): 1411-1424.

28. Li, B., Wu, S., Yang, J., Zhou, Y., & Du, M. (2012). A three-fold approach for job shop problems: A divide-and-integrate strategy with immune algorithm. *Journal of Manufacturing Systems*, *31*(2), 195-203.
29. Ruiz, Rubén, Concepción Maroto, and Javier Alcaraz. "Two new robust genetic algorithms for the flowshop scheduling problem." *Omega* 34.5 (2006): 461-476
30. Gen, M.; Tsujimura, Y.; Kubota, E., "Solving job-shop scheduling problems by genetic algorithm," in *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on* , vol.2, no., pp.1577-1582 vol.2, 2-5 Oct 1994.
31. Colin R. Reeves, A genetic algorithm for flowshop sequencing, Computers & Operations Research, Volume 22, Issue 1, January 1995, Pages 5-13, ISSN 0305-0548, http://dx.doi.org/10.1016/0305-0548(93)E0014-K.
32. Rubén Ruiz, Concepción Maroto, Javier Alcaraz, Two new robust genetic algorithms for the flowshop scheduling problem, Omega, Volume 34, Issue 5, October 2006, Pages 461-476, ISSN 0305-0483, http://dx.doi.org/10.1016/j.omega.2004.12.006.
33. Taillard, Eric. "Benchmarks for basic scheduling problems." *european journal of operational research* 64.2 (1993): 278-285.
34. Tandon, M., P. T. Cummings, and M. D. LeVan. "Flowshop sequencing with non-permutation schedules." *Computers & chemical engineering* 15.8 (1991): 601-607.
35. Demirkol, Ebru, Sanjay Mehta, and Reha Uzsoy. "Benchmarks for shop scheduling problems." *European Journal of Operational Research* 109.1 (1998): 137-141.

36. Yagmahan, Betul, and Mehmet Mutlu Yenisey. "A multi-objective ant colony system algorithm for flow shop scheduling problem." *Expert Systems with Applications* 37.2 (2010): 1361-1368.
37. Rajendran, Chandrasekharan. "Heuristics for scheduling in flowshop with multiple objectives." *European journal of operational research* 82.3 (1995): 540-555.
38. Ravindran, D., et al. "Flow shop scheduling with multiple objective of minimizing makespan and total flow time." *The international journal of advanced manufacturing technology* 25.9-10 (2005): 1007-1012.
39. Ying, Kuo-Ching, and Shih-Wei Lin. "Multi-heuristic desirability ant colony system heuristic for non-permutation flowshop scheduling problems." *The International Journal of Advanced Manufacturing Technology* 33.7-8 (2007): 793-802.
40. Bellman, Richard. "Mathematical aspects of scheduling theory." *Journal of the Society for Industrial and Applied Mathematics* 4.3 (1956): 168-205.
41. R. E. Smith and E. Smuda, “Adaptively resizing populations: Algorithm, analysis, and first results,” Complex Systems, vol. 9, no. 1, pp. 47–72, 1995.
42. T.-H. Li, C. B. Lucasius, and G. Kateman, “Optimization of calibration data with the dynamic genetic algorithm,” Analytica Chimica Acta, vol. 268, no. 1, pp. 123–134, 1992.
43. S. W. Mahfoud, “Niching Methods for Genetic Algorithms,” 1995: Illinois Genetic Algorithm Lab., Univ. Illinois.

44. J. C. Potts, T. D. Giddens, and S. B. Yadav, “The development and evaluation of an improved genetic algorithm based on migration and artificial selection,” IEEE Transactions on Systems, Man and Cybernetics, vol. 24, no. 1, pp. 73–86, 1994.
45. M. L. Mauldin, “Maintaining Diversity in Genetic Search.,” pp. 247–250, 1984.
46. J. Arabas, Z. Michalewicz, and J. Mulawka, “GAVaPS-a genetic algorithm with varying population size,” IEEE World Congress on Computational Intelligence, pp. 73–78 vol.1, 1994.
47. J. J. Grefenstette, “Optimization of Control Parameters for Genetic Algorithms,” IEEE Transactions on Systems, Man and Cybernetics, vol. 16, no. 1, pp. 122–128, 1986.
48. Herrera, Francisco, and Manuel Lozano. "Adaptation of genetic algorithm parameters based on fuzzy logic controllers." *Genetic Algorithms and Soft Computing* 8 (1996): 95-125.
49. M. A. Lee and H. Takagi, “Integrating design stages of fuzzy systems using genetic algorithms,” Proceeding 2nd IEEE International Conference. Fuzzy Systems, pp.612 -617, 1993.
50. B. H. Xu, R. C. Baird, and G. Vukovich, “Fuzzy Evolutionary Algorithms and Automatic Robot Trajectory Generation,” Methods and Applications of Intelligent Control, pp. 423–449, 1997.
51. H. Y. Xu and G. VUKOVICH, “A fuzzy genetic algorithm with effective search and optimization,” International Joint Conference on Neural Networks, vol. 3, pp. 2967–2970, 1993.

52. M. Lee and H. Takagi, “Dynamic control of genetic algorithms using fuzzy logic techniques,” Proceedings 5th International Conference genetic Algorithms, pp.76-83, 1993.
53. Pinedo, Michael. "Scheduling: theory, algorithms and systems, 1995."
54. D. H. Wolpert and W. G. Macready, “No fre lunch theorems for optimization”, IEEE Transactions on Evolutionary Computation, vol. 1, pp.67-82, 1997.
55. Campbell, Herbert G., Richard A. Dudek, and Milton L. Smith. "A heuristic algorithm for the n job, m machine sequencing problem." Management science 16.10 (1970): B-630.
56. Shirazi, Arash N., Meghan Steinhaus, and Manbir Sodhi.”Modified Genetic Algorithm Based on Human Cell Mechanisms”. Manuscript in preparation.
57. Agnetis, A., et al. "Scheduling of flexible flow lines in an automobile assembly plant." European Journal of Operational Research 97.2 (1997): 348-362.
58. Piramuthu, Selwyn, Narayan Raman, and Michael J. Shaw. "Learning-based scheduling in a flexible manufacturing flow line." Engineering Management, IEEE Transactions on 41.2 (1994): 172-182.
59. Aghezzaf, El-Houssaine, and Hendrik Van Landeghem. "An integrated model for inventory and production planning in a two-stage hybrid production system." International journal of production research 40.17 (2002): 4323-4339.
60. Grabowski, Jozef, and Jaroslaw Pempera. "Sequencing of jobs in some production system." European Journal of Operational Research 125.3 (2000): 535-550.

61. Jin, Z. H., et al. "SCHEDULING HYBRID FLOWSHOPS IN PRINTED CIRCUIT BOARD ASSEMBLY LINES*." Production and Operations Management 11.2 (2002): 216-230.
62. SHERALI, HANIF D., SUBHASH C. SARIN, and MURALIDHARAN S. KODIALAM. "Models and algorithms for a two-stage production process." Production Planning & Control 1.1 (1990): 27-39.
63. Kacem, Imed, Slim Hammadi, and Pierre Borne. "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 32.1 (2002): 1-13.
64. Gupta, Jatinder ND. "Two-stage, hybrid flowshop scheduling problem." Journal of the Operational Research Society (1988): 359-364.
65. Garey, Michael R., and David S. Johnson. "A Guide to the Theory of NP-Completeness." WH Freemann, New York (1979).
66. Kurz, Mary E., and Ronald G. Askin. "Comparing scheduling rules for flexible flow lines." International Journal of Production Economics 85.3 (2003): 371-388.
67. Brah, Shaukat A., and Luan Luan Loo. "Heuristics for scheduling in a flow shop with multiple processors." European Journal of Operational Research 113.1 (1999): 113-122.
68. Rajendran, Chandrasekharan, and Dipak Chaudhuri. "An efficient heuristic approach to the scheduling of jobs in a flowshop." European Journal of Operational Research 61.3 (1992): 318-325.

69. Moursli, Omar, and Yves Pochet. "A branch-and-bound algorithm for the hybrid flowshop." International Journal of Production Economics 64.1 (2000): 113-125.
70. Santos, D. L., J. L. Hunsucker, and D. E. Deal. "Global lower bounds for flow shops with multiple processors." European Journal of Operational Research 80.1 (1995): 112-120.
71. Fattahi, Parviz, et al. "A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations." Applied Mathematical Modelling 38.1 (2014): 119-134.
72. Gupta, Jatinder ND. "Two-stage, hybrid flowshop scheduling problem." Journal of the Operational Research Society (1988): 359-364.
73. Gupta, J. N. D., A. M. A. Hariri, and C. N. Potts. "Scheduling a two-stage hybrid flow shop with parallel machines at the first stage." Annals of Operations Research 69 (1997): 171-191.
74. Sriskandarajah, Chelliah, and Suresh P. Sethi. "Scheduling algorithms for flexible flowshops: worst and average case performance." European Journal of Operational Research 43.2 (1989): 143-160.
75. Brah, Shaukat A., and Luan Luan Loo. "Heuristics for scheduling in a flow shop with multiple processors." European Journal of Operational Research 113.1 (1999): 113-122.
76. Linn, Richard, and Wei Zhang. "Hybrid flow shop scheduling: a survey." Computers & industrial engineering 37.1 (1999): 57-61.
77. Kochhar, Sandeep, and Robert JT Morris. "Heuristic methods for flexible flow line scheduling." Journal of Manufacturing Systems 6.4 (1987): 299-314.

78. Ruiz, Rubén, and José Antonio Vázquez-Rodríguez. "The hybrid flow shop scheduling problem." European Journal of Operational Research 205.1 (2010): 1-18.
79. Voß, Stefan. "The Two—Stage Hybrid—Flowshop Scheduling Problem with Sequence—Dependent Setup Times." Operations Research in Production Planning and Control. Springer Berlin Heidelberg, 1993. 336-352.
80. Haouari, Mohamed, and Rym M'Hallah. "Heuristic algorithms for the two-stage hybrid flowshop problem." Operations research letters 21.1 (1997): 43-53.
81. Nowicki, Eugeniusz, and Czeslaw Smutnicki. "The flow shop with parallel machines: A tabu search approach." European Journal of Operational Research 106.2 (1998): 226-253.
82. Nowicki, Eugeniusz, and Czeslaw Smutnicki. "A fast taboo search algorithm for the job shop problem." Management science 42.6 (1996): 797-813.
83. Cheng, Jinliang, Yoshiyuki Karuno, and Hiroshi Kise. "A shifting bottleneck approach for a parallel-machine flowshop scheduling problem." Journal of the operations research society of Japan 44.2 (2001): 140-156.
84. Naderi, B., et al. "An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness." Expert systems with Applications 36.6 (2009): 9625-9633.
85. Jin, Zhihong, Zan Yang, and Takahiro Ito. "Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem." International Journal of Production Economics 100.2 (2006): 322-334.

86. Kurz, Mary E., and Ronald G. Askin. "Scheduling flexible flow lines with sequence-dependent setup times." European Journal of Operational Research 159.1 (2004): 66-82.
87. Zandieh, M., SMT Fatemi Ghomi, and SM Moattar Husseini. "An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times." Applied Mathematics and Computation 180.1 (2006): 111-127.
88. Mirsanei, H. S., et al. "A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times." Journal of Intelligent Manufacturing 22.6 (2011): 965-978.
89. Ruiz, Ruben, and Concepción Maroto. "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility." European Journal of Operational Research 169.3 (2006): 781-800.
90. Oğuz, Ceyda, and M. Fikret Ercan. "A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks." Journal of Scheduling 8.4 (2005): 323-351.
91. Engin, Orhan, Gülşad Ceran, and Mustafa K. Yilmaz. "An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems." Applied Soft Computing 11.3 (2011): 3056-3065.
92. Guirchoun, Samuel, Patrick Martineau, and J-C. Billaut. "Total completion time minimization in a computer system with a server and two parallel processors." Computers & Operations Research 32.3 (2005): 599-611.
93. Shirazi, Arash N., Meghan Steinhaus, and Manbir Sodhi.”Modified Genetic Algorithm Based on Human Cell Mechanisms”. Manuscript in preparation.

94. Shirazi, Arash N., Meghan Steinhaus, and Manbir Sodhi.”Scheduling Flow Line by Fuzzy Cell Genetic Algorithm”. Manuscript in preparation.
95. Brooke, A. Kendrick, and A. D Meeraus. GAMS release 2.25; a user's guide. No. 519.76 B872. GAMS Development Corporation, Washington, DC (EUA), 1996.
96. A. Scholl, N. Boysen, and M. Fliedner, “The sequence-dependent assembly line balancing problem” OR Spectrum, vol. 30,pp. 579-609, March 2008.
97. A. SHTTJB, “The effect of incompletion cost on line balancing with multiple manning of work stations.” THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH vol. 2 , pp. 235-245,1984.
98. A. Scholl. Balancing and Sequencing of Assembly Lines. Heidelberg; New York: Physica-Verlag, 1999.
99. A.E. Owen, Assembly with Robots. Kogan Page Ltd, 1985.
100. Spieckermann, S., K. Gutenschwager, H. Heinzel, and S. Voß. 2000. “Simulation based optimization in the automotive industry - A case study on body shop design.”Simulation 75 (5-6): 276–286.
101. Boysen, N., M. Fliedner, and A. Scholl. 2008. “Assembly line balancing: Which model to use when?.” International Journal of Production Economics 111 (2): 509–528.
102. Helgeson, W. B., M. E. Salveson, and W. W. Smith. "How to balance an assembly line." Management Report 7 (1954).
103. Salveson, Melvin E. "The assembly line balancing problem." Journal of Industrial Engineering 6, vol. 3 , p.p. 18-25,1955.

104. Bowman, Edward H. "Assembly-line balancing by linear programming." Operations Research 8, no. 3 (1960): 385-389.

105. Held, Michael, Richard M. Karp, and Richard Shareshian. "Assembly-line balancing—dynamic programming with precedence constraints." Operations Research 11, no. 3 (1963): 442-459.

106. Karp, Richard M. Reducibility among combinatorial problems. Springer US, 1972.

107. Gen, M.,& Cheng, R.(200). Genetic Algorithm and Engineering Optimization, New York:John Wiley & Sons.

108. Rubinovitz, J., and Gregory Levitin. "Genetic algorithm for assembly line balancing." International Journal of Production Economics 41.1 (1995): 343-354.

109. Kim, Yeo Keun, Yong Ju Kim, and Yeongho Kim. "Genetic algorithms for assembly line balancing with various objectives." Computers & Industrial Engineering 30.3 (1996): 397-409.

110. Leu, Y. Y., Matheson, L. A., & Rees, L. P. (1994). Assembly Line Balancing Using Genetic Algorithms with Heuristic-Generated Initial Populations and Multiple Evaluation Criteria*. Decision Sciences, 25(4), 581-605.

111. Sabuncuoglu, I., Erel, E., & Tanyer, M. (2000). Assembly line balancing using genetic algorithms. Journal of intelligent manufacturing, 11(3), 295-310.

112. Haq, A. N., Rengarajan, K., & Jayaprakash, J. (2006). A hybrid genetic algorithm approach to mixed-model assembly line balancing. The International Journal of Advanced Manufacturing Technology, 28(3-4), 337-341.

113. Levitin, Gregory, Jacob Rubinovitz, and Boris Shnits. "A genetic algorithm for robotic assembly line balancing." European Journal of Operational Research 168.3 (2006): 811-825.

114. Tasan, Seren Ozmehmet, and Semra Tunali. "A review of the current applications of genetic algorithms in assembly line balancing." Journal of intelligent manufacturing 19.1 (2008): 49-69.

115. Rekiek, Brahim, et al. "State of art of optimization methods for assembly line design." Annual Reviews in Control 26.2 (2002): 163-174.

116. Bartholdi, J. J. "Balancing two-sided assembly lines: a case study." THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 31, no. 10 (1993): 2447-2461.

117. Gen, Mitsuo, Runwei Cheng, and Lin Lin. Network models and optimization: Multiobjective genetic algorithm approach. Springer, 2008.

118. Bartholdi, J. J. "Balancing two-sided assembly lines: a case study." THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 31, no. 10 (1993): 2447-2461.

119. Augugliaro, Antonino, Luigi Dusonchet, and Eleonora Riva Sanseverino. "Evolving non-dominated solutions in multiobjective service restoration for automated distribution networks." Electric Power Systems Research 59.3 (2001): 185-195.

120. Kahan, Tomer, Yossi Bukchin, Roland Menassa, and Irad Ben-Gal. "Backup strategy for robots' failures in an automotive assembly system." International Journal of Production Economics 120, no. 2 (2009): 315-326.

121. Shin, Frank, Bala Ram, Aman Gupta, Xuefeng Yu, and Roland Menassa. "A decision tool for assembly line breakdown action." In Proceedings of the 36th conference on Winter simulation, pp. 1122-1127. Winter Simulation Conference, 2004.

122. Müller, Christoph, Thomas S. Spengler, and Manbir S. Sodhi. "Robust Flowline Design for Automotive Body Shops." IIE Annual Conference. Proceedings. Institute of Industrial Engineers-Publisher, 2014.

123. Gupta, Ashish, Gopalakrishna K. Shastry, Narahari K. Hunsur, Wayne Cai, and Robert Tilove. "An Approach for Automated and Optimized Selection of Weldguns for Spot Welding in Automotive Body Shop." Transactions of the ASME-B-Journ Manufacturing Science Engineering 134, no. 3 (2012): 034501.